Package: slimr (via r-universe)

November 15, 2024

Title Create, Run and Post-Process 'SLiM' Population Genetics Forward Simulations

Version 0.4.0.9000

Description Lets you write 'SLiM' scripts (population genomics simulation) using your favourite R IDE, using a syntax as close as possible to the original 'SLiM' language. It offer many tools to manipulate those scripts, as well as run them in the 'SLiM' software from R, as well as capture and post-process their output, after or even during a simulation.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

URL https://rdinnager.github.io/slimr, https://rdinnager.github.io/slimr/index.html, https://github.com/rdinnager/slimr

BugReports https://github.com/rdinnager/slimr/issues

RoxygenNote 7.2.3

Language en-GB

Depends R (>= 2.10)

Imports stringr, processx, readr, glue, magrittr, progress, dplyr, rlang (>= 0.1.2), purrr, prettycode, vctrs, tidyr, zeallot, utils, stats, codetools

Suggests ape, covr, testthat (>= 2.1.0), spelling, knitr, rmarkdown, roxygen2, clipr, ggplot2, gganimate, grDevices, fansi, printr, rstudioapi, crayon, adegenet, methods, furrr, future, Biostrings, raster, withr, paletteer, reticulate

VignetteBuilder knitr

Config/testthat/edition 3

Config/pak/sysreqs libicu-dev libx11-dev

Repository https://rdinnager.r-universe.dev

RemoteUrl https://github.com/rdinnager/slimr

RemoteRef 0.4.0

RemoteSha ddf6789db5f3fdaa78d653f20d0fe5b8bc977709

Contents

`add` *SLiM method add*

Description

Documentation for SLiM function `add`, which is a method of the SLiM class `SpatialMap`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
add(x)
```

Arguments

`x`	An object of type integer or float or SpatialMap object. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 713.

Adds x to the spatial map. One possibility is that x is a singleton integer or float value; in this case, x is added to each grid value of the target spatial map. Another possibility is that x is an integer or float vector/matrix/array of the same dimensions as the target spatial map's grid; in this case, each value of x is added to the corresponding grid value of the target spatial map. The third possibility is that x is itself a (singleton) spatial map; in this case, each grid value of x is added to the corresponding grid value of the target spatial map (and thus the two spatial maps must match in their spatiality, their spatial bounds, and their grid dimensions). The target spatial map is returned, to allow easy chaining of operations.

Value

An object of type SpatialMap object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other SpatialMap: SM, blend(), changeColors(), changeValues(), divide(), exp(), gridValues(), interpolate(), mapColor(), mapImage(), mapValue(), multiply(), power(), range(), rescale(), sampleImprovedNearbyPoint(), sampleNearbyPoint(), smooth(), subtract()

---

addCloned *SLiM method addCloned*

Description

Documentation for SLiM function addCloned, which is a method of the SLiM class Subpopulation. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
addCloned(parent, count, defer)
```

Arguments

parent	An object of type Individual object. Must be of length 1 (a singleton). See details for description.
count	An object of type integer. Must be of length 1 (a singleton). The default value is 1. See details for description.
defer	An object of type logical. Must be of length 1 (a singleton). The default value is F. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 732.

Generates a new offspring individual from the given parent by clonal reproduction, queues it for addition to the target subpopulation, and returns it. The new offspring will not be visible as a member of the target subpopulation until the end of the offspring generation tick cycle stage. The subpopulation of parent will be used to locate applicable mutation() and modifyChild() callbacks governing the generation of the offspring individual. Beginning in SLiM 4.1, the count parameter dictates how many offspring will be generated (previously, exactly one offspring was generated). Each offspring is generated independently, based upon the given parameters. The returned vector contains all generated offspring, except those that were rejected by a modifyChild() callback. If all offspring are rejected, object<Individual>(0) is returned, which is a zero-length object vector of class Individual;

note that this is a change in behavior from earlier versions, which would return NULL. Beginning in SLiM 4.1, passing T for defer will defer the generation of the genomes of the produced offspring until the end of the reproduction phase. Genome generation can only be deferred if there are no active mutation() callbacks; otherwise, an error will result. Furthermore, when genome generation is deferred the mutations of the genomes of the generated offspring may not be accessed until reproduction is complete (whether from a modifyChild() callback or otherwise). There is little or no advantage to deferring genome generation at this time (it is in place for future expansion); the default of F for defer is generally preferable since it has fewer restrictions. Also beginning in SLiM 4.1, in spatial models the spatial position of the offspring will be inherited (i.e., copied) from parent; more specifically, the x property will be inherited in all spatial models (1D/ 2D/3D), the y property in 2D/3D models, and the z property in 3D models. Properties not inherited will be left uninitialized, as they were prior to SLiM 4.1. The parent's spatial position is probably not desirable in itself; the intention here is to make it easy to model the natal dispersal of all the new offspring for a given tick with a single vectorized call to pointDeviated(). Note that this method is only for use in nonWF models. See addCrossed() for further general notes on the addition of new offspring individuals.

Value

An object of type Individual object.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Subpopulation: P, addCrossed(), addEmpty(), addRecombinant(), addSelfed(), addSpatialMap(), cachedFitness(), configureDisplay(), defineSpatialMap(), outputMSSample(), outputSample(), outputVCFSample(), pointDeviated(), pointInBounds(), pointPeriodic(), pointReflected(), pointStopped(), pointUniform(), removeSpatialMap(), removeSubpopulation(), sampleIndividuals(), setCloningRate(), setMigrationRates(), setSelfingRate(), setSexRatio(), setSpatialBounds(), setSubpopulationSize(), spatialMapColor(), spatialMapImage(), spatialMapValue(), subsetIndividuals(), takeMigrants()

`addCrossed` *SLiM method addCrossed*

Description

Documentation for SLiM function `addCrossed`, which is a method of the SLiM class `Subpopulation`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
addCrossed(parent1, parent2, sex, count, defer)
```

Arguments

`parent1`	An object of type Individual object. Must be of length 1 (a singleton). See details for description.
`parent2`	An object of type Individual object. Must be of length 1 (a singleton). See details for description.
`sex`	An object of type null or float or string. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`count`	An object of type integer. Must be of length 1 (a singleton). The default value is `1`. See details for description.
`defer`	An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 733.

Generates a new offspring individual from the given parents by biparental sexual reproduction, queues it for addition to the target subpopulation, and returns it. The new offspring will not be visible as a member of the target subpopulation until the end of the offspring generation tick cycle stage. Attempting to use a newly generated offspring individual as a mate, or to reference it as a member of the target subpopulation in any other way, will result in an error. In most models the returned individual is not used, but it is provided for maximal generality and flexibility. The new offspring individual is generated from parent1 and parent2 by crossing them. In sexual models parent1 must be female and parent2 must be male; in hermaphroditic models, parent1 and parent2 are unrestricted. If parent1 and parent2 are the same individual in a hermaphroditic model, that parent self-fertilizes, or "selfs", to generate the offspring sexually (note this is not the same as clonal reproduction). Such selfing is considered "incidental" by addCrossed(), however; if the preventIncidentalSelfing flag of initializeSLiMOptions() is T, supplying the same individual for parent1 and parent2 is an error (you must check for and prevent incidental selfing if you set that flag in a nonWF model). If non-incidental selfing is desired, addSelfed() should be used instead. The

sex parameter specifies the sex of the offspring. A value of NULL means "make the default choice"; in non-sexual models it is the only legal value for sex, and does nothing, whereas in sexual models it causes male or female to be chosen with equal probability. A value of "M" or "F" for sex specifies that the offspring should be male or female, respectively. Finally, a float value from 0.0 to 1.0 for sex provides the probability that the offspring will be male; a value of 0.0 will produce a female, a value of 1.0 will produce a male, and for intermediate values SLiM will draw the sex of the offspring randomly according to the specified probability. Unless you wish the bias the sex ratio of offspring, the default value of NULL should generally be used. Note that any defined, active, and applicable recombination(), mutation(), and modifyChild() callbacks will be called as a side effect of calling this method, before this method even returns. For recombination() and mutation() callbacks, the subpopulation of the parent that is generating a given gamete is used; for modifyChild() callbacks the situation is more complex. In most biparental mating events, parent1 and parent2 will belong to the same subpopulation, and modifyChild() callbacks for that subpopulation will be used, just as in WF models. In certain models (such as models of pollen flow and broadcast spawning), however, biparental mating may occur between parents that are not from the same subpopulation; that is legal in nonWF models, and in that case, modifyChild() callbacks for the subpopulation of parent1 are used (since that is the maternal parent). If the modifyChild() callback process results in rejection of the proposed child (see section 26.5), a new offspring individual is not be generated. To force the generation of an offspring individual from a given pair of parents, you could loop until addCrossed() succeeds, but note that if your modifyChild() callback rejects all proposed children from those particular parents, your model will then hang, so care must be taken with this approach. Usually, nonWF models do not force generation of offspring in this manner; rejection of a proposed offspring by a modifyChild() callback typically represents a phenomenon such as post-mating reproductive isolation or lethal genetic incompatibilities that would reduce the expected litter size, so the default behavior is typically desirable. Beginning in SLiM 4.1, the count parameter dictates how many offspring will be generated (previously, exactly one offspring was generated). Each offspring is generated independently, based upon the given parameters. The returned vector contains all generated offspring, except those that were rejected by a modifyChild() callback. If all offspring are rejected, object<Individual>(0) is returned, which is a zero-length object vector of class Individual; note that this is a change in behavior from earlier versions, which would return NULL. Beginning in SLiM 4.1, passing T for defer will defer the generation of the genomes of the produced offspring until the end of the reproduction phase. Genome generation can only be deferred if there are no active mutation() or recombination() callbacks; otherwise, an error will result. Furthermore, when genome generation is deferred the mutations of the genomes of the generated offspring may not be accessed until reproduction is complete (whether from a modifyChild() callback or otherwise). There is little or no advantage to deferring genome generation at this time (it is in place for future expansion); the default of F for defer is generally preferable since it has fewer restrictions. Also beginning in SLiM 4.1, in spatial models the spatial position of the offspring will be inherited (i.e., copied) from parent1; more specifically, the x property will be inherited in all spatial models (1D/2D/3D), the y property in 2D/3D models, and the z property in 3D models. Properties not inherited will be left uninitialized, as they were prior to SLiM 4.1. The parent's spatial position is probably not desirable in itself; the intention here is to make it easy to model the natal dispersal of all the new offspring for a given tick with a single vectorized call to pointDeviated(). Note that this method is only for use in nonWF models, in which offspring generation is managed manually by the model script; in

such models, addCrossed() must be called only from reproduction() callbacks, and may not be called at any other time. In WF models, offspring generation is managed automatically by the SLiM core.

Value

An object of type Individual object.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Subpopulation: P, addCloned(), addEmpty(), addRecombinant(), addSelfed(), addSpatialMap(), cachedFitness(), configureDisplay(), defineSpatialMap(), outputMSSample(), outputSample(), outputVCFSample(), pointDeviated(), pointInBounds(), pointPeriodic(), pointReflected(), pointStopped(), pointUniform(), removeSpatialMap(), removeSubpopulation(), sampleIndividuals(), setCloningRate(), setMigrationRates(), setSelfingRate(), setSexRatio(), setSpatialBounds(), setSubpopulationSize(), spatialMapColor(), spatialMapImage(), spatialMapValue(), subsetIndividuals(), takeMigrants()

---

addCustomColumn *SLiM method addCustomColumn*

Description

Documentation for SLiM function addCustomColumn, which is a method of the SLiM class LogFile. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
addCustomColumn(columnName, source, context)
```

Arguments

`columnName`	An object of type string or string or any. Must be of length 1 (a singleton). See details for description.
`source`	An object of type string or string or any. Must be of length 1 (a singleton). See details for description.
`context`	An object of type string or string or any. The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 701.

Adds a new data column with its name provided by columnName. The value for the column, when a given row is generated, will be produced by the code supplied in source, which is expected to return either NULL (which will write out NA), or a singleton value of any non-object type. The context parameter will be set up as a pseudo-parameter, named context, when source is called, allowing the same source code to be used to generate values for multiple data columns; you might, for example, provide the particular Subpopulation object here that you wish source to use for its calculations. This is optional; if the default value of NULL is used, then context will be NULL when source is called. See addMeanSDColumns() for a useful variant.

Value

An object of type void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or logical.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other LogFile: LF, addCycleStage(), addCycle(), addKeysAndValuesFrom(), addMeanSDColumns(), addPopulationSexRatio(), addPopulationSize(), addSubpopulationSexRatio(), addSubpopulationSize(), addSuppliedColumn(), addTick(), clearKeysAndValues(), flush(), logRow(), setFilePath(), setLogInterval(), setSuppliedValue(), setValue(), willAutolog()

`addCycle` *SLiM method addCycle*

Description

Documentation for SLiM function `addCycle`, which is a method of the SLiM class `LogFile`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
addCycle(species)
```

Arguments

`species`	An object of type null or Species object. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 701.

Adds a new data column that provides the cycle counter for species (the same as the value of the cycle property of that species). In single-species models, species may be NULL to indicate that single species. The column will simply be named cycle in single-species models; an underscore and the name of the species will be appended in multispecies models.

Value

An object of type void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or logical.

Copyright

Author(s)

Benjamin C Haller (`<bhaller@benhaller.com>`) and Philipp W Messer (`<messer@cornell.edu>`)

See Also

Other LogFile: LF, addCustomColumn(), addCycleStage(), addKeysAndValuesFrom(), addMeanSDColumns(), addPopulationSexRatio(), addPopulationSize(), addSubpopulationSexRatio(), addSubpopulationSize(), addSuppliedColumn(), addTick(), clearKeysAndValues(), flush(), logRow(), setFilePath(), setLogInterval(), setSuppliedValue(), setValue(), willAutolog()

`addCycleStage` *SLiM method addCycleStage*

Description

Documentation for SLiM function `addCycleStage`, which is a method of the SLiM class `LogFile`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
addCycleStage(void)
```

Arguments

`void`	An object of type . See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 701.

Adds a new data column that provides the cycle stage, named cycle_stage. The stage is provided as a string, and will typically be "first", "early", "late", or "end" (the latter used for the point in time at which end-of-tick automatic logging occurs). Other possible values are discussed in the documentation for the cycleStage property of Community, which this column reflects.

Value

An object of type void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or logical.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other LogFile: LF, addCustomColumn(), addCycle(), addKeysAndValuesFrom(), addMeanSDColumns(), addPopulationSexRatio(), addPopulationSize(), addSubpopulationSexRatio(), addSubpopulationSize(), addSuppliedColumn(), addTick(), clearKeysAndValues(), flush(), logRow(), setFilePath(), setLogInterval(), setSuppliedValue(), setValue(), willAutolog()

`addEmpty` *SLiM method addEmpty*

Description

Documentation for SLiM function `addEmpty`, which is a method of the SLiM class `Subpopulation`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
addEmpty(sex, genome1Null, genome2Null, count)
```

Arguments

`sex`	An object of type null or float or string. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`genome1Null`	An object of type null or logical. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`genome2Null`	An object of type null or logical. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`count`	An object of type integer. Must be of length 1 (a singleton). The default value is `1`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 734.

Generates a new offspring individual with empty genomes (i.e., containing no mutations), queues it for addition to the target subpopulation, and returns it. The new offspring will not be visible as a member of the target subpopulation until the end of the offspring generation tick cycle stage. No recombination() or mutation() callbacks will be called. The target subpopulation will be used to locate applicable modifyChild() callbacks governing the generation of the offspring individual (unlike the other addX() methods, because there is no parental individual to reference). The offspring is considered to have no parents

for the purposes of pedigree tracking. The sex parameter is treated as in addCrossed(). By default - when genome1Null and genome2Null are both NULL - null genomes will be generated instead of empty genomes only in sex-chromosome simulations, where the sex chromosome that is not being simulated is represented by a null genome; otherwise, empty genomes rather than null genomes will be created. This default behavior can be changed by passing T or F for genome1Null or genome2Null, which will force the corresponding offspring genome to be null (T) or non-null (F). The behavior in sex-chromosome simulations cannot be changed, since the presence of null genomes there is dictated by sex, but T or F may be passed as long as it matches what SLiM would do anyway. In all other simulations there is little point in passing F (since that would be the default behavior anyway), but passing T can be used to make one or both genomes be null genomes, which can be useful for, e.g., modeling haploids (for which, by convention, the second genome is usually a null genome in SLiM). Beginning in SLiM 4.1, the count parameter dictates how many offspring will be generated (previously, exactly one offspring was generated). Each offspring is generated independently, based upon the given parameters. The returned vector contains all generated offspring, except those that were rejected by a modifyChild() callback. If all offspring are rejected, object<Individual>(0) is returned, which is a zero-length object vector of class Individual; note that this is a change in behavior from earlier versions, which would return NULL. Note that this method is only for use in nonWF models. See addCrossed() for further general notes on the addition of new offspring individuals.

Value

An object of type Individual object.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Subpopulation: P, addCloned(), addCrossed(), addRecombinant(), addSelfed(), addSpatialMap(), cachedFitness(), configureDisplay(), defineSpatialMap(), outputMSSample(), outputSample(), outputVCFSample(), pointDeviated(), pointInBounds(), pointPeriodic(), pointReflected(), pointStopped(), pointUniform(), removeSpatialMap(), removeSubpopulation(), sampleIndividuals(), setCloningRate(), setMigrationRates(), setSelfingRate(), setSexRatio(), setSpatialBounds(), setSubpopulationSize(), spatialMapColor(), spatialMapImage(), spatialMapValue(), subsetIndividuals(), takeMigrants()

`addKeysAndValuesFrom` *SLiM method addKeysAndValuesFrom*

Description

Documentation for SLiM function `addKeysAndValuesFrom`, which is a method of the SLiM class `LogFile`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
addKeysAndValuesFrom(source)
```

Arguments

`source`	An object of type . Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 702.

This Dictionary method has an override in LogFile to make it illegal to call, since LogFile manages its Dictionary entries.

Value

An object of type void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or logical.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other LogFile: `LF`, `addCustomColumn()`, `addCycleStage()`, `addCycle()`, `addMeanSDColumns()`, `addPopulationSexRatio()`, `addPopulationSize()`, `addSubpopulationSexRatio()`, `addSubpopulationSize()`, `addSuppliedColumn()`, `addTick()`, `clearKeysAndValues()`, `flush()`, `logRow()`, `setFilePath()`, `setLogInterval()`, `setSuppliedValue()`, `setValue()`, `willAutolog()`

`addMeanSDColumns` *SLiM method addMeanSDColumns*

Description

Documentation for SLiM function `addMeanSDColumns`, which is a method of the SLiM class `LogFile`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
addMeanSDColumns(columnName, source, context)
```

Arguments

`columnName`	An object of type string or string or any. Must be of length 1 (a singleton). See details for description.
`source`	An object of type string or string or any. Must be of length 1 (a singleton). See details for description.
`context`	An object of type string or string or any. The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 702.

Adds two new data columns with names of columnName_mean and columnName_sd. When a given row is generated, the code supplied in source is expected to return either a zero-length vector of any type including NULL (which will write out NA to both columns), or a non-zero-length vector of integer or float values. In the latter case, the result vector will be summarized in the two columns by its mean and standard deviation respectively. If the result vector has exactly one value, the standard deviation will be written as NA. The context parameter is set up as a pseudo-parameter when source is called, as described in addCustomColumn().

Value

An object of type void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or logical.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other LogFile: LF, addCustomColumn(), addCycleStage(), addCycle(), addKeysAndValuesFrom(), addPopulationSexRatio(), addPopulationSize(), addSubpopulationSexRatio(), addSubpopulationSize(), addSuppliedColumn(), addTick(), clearKeysAndValues(), flush(), logRow(), setFilePath(), setLogInterval(), setSuppliedValue(), setValue(), willAutolog()

---

addMutations *SLiM method addMutations*

Description

Documentation for SLiM function addMutations, which is a method of the SLiM class Genome. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
addMutations(mutations)
```

Arguments

mutations	An object of type Mutation object. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 670.

Add the existing mutations in mutations to the genome, if they are not already present (if they are already present, they will be ignored), and if the addition is not prevented by the mutation stacking policy (see the mutationStackPolicy property of MutationType, section 25.11.1). Calling this will normally affect the fitness values calculated toward the end of the current tick; if you want current fitness values to be affected, you can call the Species method recalculateFitness() - but see the documentation of that method for caveats. Note that in nonWF models that use tree-sequence recording, mutations cannot be added to an individual after the tick in which the individual is created (i.e., when the age of the individual is greater than 0), to prevent the possibility of inconsistencies in the recorded tree sequence.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Genome: G, addNewDrawnMutation(), addNewMutation(), containsMarkerMutation(), containsMutations(), countOfMutationsOfType(), mutationCountsInGenomes(), mutationFrequenciesInGe mutationsOfType(), nucleotides(), outputMS(), outputVCF(), output(), positionsOfMutationsOfType(), readFromMS(), readFromVCF(), removeMutations(), sumOfMutationsOfType()

---

addNewDrawnMutation *SLiM method addNewDrawnMutation*

Description

Documentation for SLiM function addNewDrawnMutation, which is a method of the SLiM class Genome. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
addNewDrawnMutation(
  mutationType,
  position,
  originTick,
  originSubpop,
  nucleotide
)
```

Arguments

mutationType	An object of type integer or MutationType object. See details for description.
position	An object of type integer. See details for description.
originTick	An object of type null or integer. The default value is NULL. See details for description.

`originSubpop` An object of type null or integer or Subpopulation object. The default value is `NULL`. See details for description.

`nucleotide` An object of type null or integer or string. The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 670.

Add new mutations to the target genome(s) with the specified mutationType (specified by the MutationType object or by integer identifier), position, originTick (which may be NULL, the default, to specify the current tick; otherwise, beginning in SLiM 3.5, it must be equal to the current tick anyway, as other uses of this property have been deprecated), and originSubpop (specified by the Subpopulation object or by integer identifier, or by NULL, the default, to specify the subpopulation to which the first target genome belongs). If originSubpop is supplied as an integer, it is intentionally not checked for validity; you may use arbitrary values of originSubpop to ”tag” the mutations that you create (see section 25.10.1). The selection coefficients of the mutations are drawn from their mutation types; addNewMutation() may be used instead if you wish to specify selection coefficients. In non-nucleotide-based models, mutationType will always be a non-nucleotide-based mutation type, and so nucleotide must be NULL (the default). In a nucleotide-based model, mutationType might still be non-nucleotide-based (in which case nucleotide must still be NULL), or mutationType might be nucleotide-based, in which case a non-NULL value must be supplied for nucleotide, specifying the nucleotide(s) to be associated with the new mutation(s). Nucleotides may be specified with string values (”A”, ”C”, ”G”, or ”T”), or with integer values (A=0, C=1, G=2, T=3). If a nucleotide mutation already exists at the mutating position, it is replaced automatically in accordance with the stacking policy for nucleotidebased mutation types. No check is performed that a new mutation’s nucleotide differs from the ancestral sequence, or that its selection coefficient is consistent with other mutations that may already exist at the given position with the same nucleotide; model consistency is the responsibility of the model. Beginning in SLiM 2.5 this method is vectorized, so all of these parameters may be singletons (in which case that single value is used for all mutations created by the call) or non-singleton vectors (in which case one element is used for each corresponding mutation created). Nonsingleton parameters must match in length, since their elements need to be matched up oneto- one. The new mutations created by this method are returned, even if their actual addition is prevented by the mutation stacking policy (see the mutationStackPolicy property of MutationType, section 25.11.1). However, the order of the mutations in the returned vector is not guaranteed to be the same as the order in which the values are specified in parameter vectors, unless the position parameter is specified in ascending order. In other words, presorting the parameters to this method into ascending order by position, using order() and subsetting, will guarantee that the order of the returned vector of mutations corresponds to the order of elements in the parameters to this method; otherwise, no such guarantee exists. Beginning in SLiM 2.1, this is a class method, not an instance method. This means that it does not get multiplexed out to all of the elements of the receiver (which would add a different new mutation to each element); instead, it is performed as a single operation, adding the same new mutation objects to all of the elements of the receiver. Before SLiM 2.1, to add the same mutations to multiple genomes, it was necessary to call addNewDrawnMutation() on one of the genomes, and then add the returned Mutation object to all of the other genomes using addMutations().

That is not necessary in SLiM 2.1 and later, because of this change (although doing it the old way does no harm and produces identical behavior). Pre-2.1 code that actually relied upon the old multiplexing behavior will no longer work correctly (but this is expected to be an extremely rare pattern of usage). Before SLiM 4, this method also took an originGeneration parameter. This was deprecated (the origin generation was then required to be equal to the current generation, for internal consistency), and was removed in SLiM 4. Calling this will normally affect the fitness values calculated at the end of the current tick (but not sooner); if you want current fitness values to be affected, you can call the Species method recalculateFitness() - but see the documentation of that method for caveats. Note that in nonWF models that use tree-sequence recording, mutations cannot be added to an individual after the tick in which the individual is created (i.e., when the age of the individual is greater than 0), to prevent the possibility of inconsistencies in the recorded tree sequence.

Value

An object of type Mutation object.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Genome: G, addMutations(), addNewMutation(), containsMarkerMutation(), containsMutations(), countOfMutationsOfType(), mutationCountsInGenomes(), mutationFrequenciesInGe
mutationsOfType(), nucleotides(), outputMS(), outputVCF(), output(), positionsOfMutationsOfType(), readFromMS(), readFromVCF(), removeMutations(), sumOfMutationsOfType()

---

addNewMutation *SLiM method addNewMutation*

Description

Documentation for SLiM function addNewMutation, which is a method of the SLiM class Genome. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
addNewMutation(
  mutationType,
  selectionCoeff,
  position,
  originTick,
  originSubpop,
  nucleotide
)
```

Arguments

`mutationType`	An object of type integer or MutationType object. See details for description.
`selectionCoeff`	An object of type numeric. See details for description.
`position`	An object of type integer. See details for description.
`originTick`	An object of type null or integer. The default value is `NULL`. See details for description.
`originSubpop`	An object of type null or integer or Subpopulation object. The default value is `NULL`. See details for description.
`nucleotide`	An object of type null or integer or string. The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 671.

Add new mutations to the target genome(s) with the specified mutationType (specified by the MutationType object or by integer identifier), selectionCoeff, position, originTick (which may be NULL, the default, to specify the current tick; otherwise, beginning in SLiM 3.5, it must be equal to the current tick anyway, as other uses of this property have been deprecated), and originSubpop (specified by the Subpopulation object or by integer identifier, or by NULL, the default, to specify the subpopulation to which the first target genome belongs). If originSubpop is supplied as an integer, it is intentionally not checked for validity; you may use arbitrary values of originSubpop to ”tag” the mutations that you create (see section 25.10.1). The addNewDrawnMutation() method may be used instead if you wish selection coefficients to be drawn from the mutation types of the mutations. In non-nucleotide-based models, mutationType will always be a non-nucleotide-based mutation type, and so nucleotide must be NULL (the default). In a nucleotide-based model, mutationType might still be non-nucleotide-based (in which case nucleotide must still be NULL), or mutationType might be nucleotide-based, in which case a non-NULL value must be supplied for nucleotide, specifying the nucleotide(s) to be associated with the new mutation(s). Nucleotides may be specified with string values (”A”, ”C”, ”G”, or ”T”), or with integer values (A=0, C=1, G=2, T=3). If a nucleotide mutation already exists at the mutating position, it is replaced automatically in accordance with the stacking policy for nucleotidebased mutation types. No check is performed that a new mutation’s nucleotide differs from the ancestral sequence, or that its selection coefficient is consistent with other

mutations that may already exist at the given position with the same nucleotide; model consistency is the responsibility of the model. The new mutations created by this method are returned, even if their actual addition is prevented by the mutation stacking policy (see the mutationStackPolicy property of MutationType, section 25.11.1). However, the order of the mutations in the returned vector is not guaranteed to be the same as the order in which the values are specified in parameter vectors, unless the position parameter is specified in ascending order. In other words, presorting the parameters to this method into ascending order by position, using order() and subsetting, will guarantee that the order of the returned vector of mutations corresponds to the order of elements in the parameters to this method; otherwise, no such guarantee exists. Beginning in SLiM 2.1, this is a class method, not an instance method. This means that it does not get multiplexed out to all of the elements of the receiver (which would add a different new mutation to each element); instead, it is performed as a single operation, adding the same new mutation object to all of the elements of the receiver. Before SLiM 2.1, to add the same mutation to multiple genomes, it was necessary to call addNewMutation() on one of the genomes, and then add the returned Mutation object to all of the other genomes using addMutations(). That is not necessary in SLiM 2.1 and later, because of this change (although doing it the old way does no harm and produces identical behavior). Pre-2.1 code that actually relied upon the old multiplexing behavior will no longer work correctly (but this is expected to be an extremely rare pattern of usage). Before SLiM 4, this method also took a originGeneration parameter. This was deprecated (the origin generation was then required to be equal to the current generation, for internal consistency), and was removed in SLiM 4. Calling this will normally affect the fitness values calculated at the end of the current tick (but not sooner); if you want current fitness values to be affected, you can call the Species method recalculateFitness() - but see the documentation of that method for caveats. Note that in nonWF models that use tree-sequence recording, mutations cannot be added to an individual after the tick in which the individual is created (i.e., when the age of the individual is greater than 0), to prevent the possibility of inconsistencies in the recorded tree sequence.

Value

An object of type Mutation object.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Genome: G, addMutations(), addNewDrawnMutation(), containsMarkerMutation(), containsMutations(), countOfMutationsOfType(), mutationCountsInGenomes(), mutationFrequenciesInGe

mutationsOfType(), nucleotides(), outputMS(), outputVCF(), output(), positionsOfMutationsOfType(), readFromMS(), readFromVCF(), removeMutations(), sumOfMutationsOfType()

---

addPopulationSexRatio

SLiM method addPopulationSexRatio

Description

Documentation for SLiM function addPopulationSexRatio, which is a method of the SLiM class LogFile. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
addPopulationSexRatio(species)
```

Arguments

species	An object of type null or Species object. Must be of length 1 (a singleton). The default value is NULL. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 702.

Adds a new data column that provides the population sex ratio M:(M+F) for species. In singlespecies models, species may be NULL to indicate that single species. The column will simply be named sex_ratio in single-species models; an underscore and the name of the species will be appended in multispecies models. If the species is hermaphroditic, NA will be written.

Value

An object of type void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or logical.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other LogFile: LF, addCustomColumn(), addCycleStage(), addCycle(), addKeysAndValuesFrom(), addMeanSDColumns(), addPopulationSize(), addSubpopulationSexRatio(), addSubpopulationSize(), addSuppliedColumn(), addTick(), clearKeysAndValues(), flush(), logRow(), setFilePath(), setLogInterval(), setSuppliedValue(), setValue(), willAutolog()

`addPopulationSize` *SLiM method addPopulationSize*

Description

Documentation for SLiM function `addPopulationSize`, which is a method of the SLiM class LogFile. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
addPopulationSize(species)
```

Arguments

`species`	An object of type null or Species object. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 702.

Adds a new data column that provides the total population size for species. In single-species models, species may be NULL to indicate that single species. The column will simply be named num_individuals in single-species models; an underscore and the name of the species will be appended in multispecies models.

Value

An object of type void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or logical.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other LogFile: LF, addCustomColumn(), addCycleStage(), addCycle(), addKeysAndValuesFrom(), addMeanSDColumns(), addPopulationSexRatio(), addSubpopulationSexRatio(), addSubpopulationSize(), addSuppliedColumn(), addTick(), clearKeysAndValues(), flush(), logRow(), setFilePath(), setLogInterval(), setSuppliedValue(), setValue(), willAutolog()

---

addRecombinant *SLiM method addRecombinant*

Description

Documentation for SLiM function addRecombinant, which is a method of the SLiM class Subpopulation. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
addRecombinant(
  strand1,
  strand2,
  breaks1,
  strand3,
  strand4,
  breaks2,
  sex,
  parent1,
  parent2,
  randomizeStrands,
  count,
  defer
)
```

Arguments

strand1	An object of type null or Genome object. Must be of length 1 (a singleton). See details for description.
strand2	An object of type null or Genome object. Must be of length 1 (a singleton). See details for description.
breaks1	An object of type null or integer. See details for description.
strand3	An object of type null or Genome object. Must be of length 1 (a singleton). See details for description.

`strand4` An object of type null or Genome object. Must be of length 1 (a singleton). See details for description.

`breaks2` An object of type null or integer. See details for description.

`sex` An object of type null or float or string. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

`parent1` An object of type null or Individual object. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

`parent2` An object of type null or Individual object. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

`randomizeStrands` An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

`count` An object of type integer. Must be of length 1 (a singleton). The default value is `1`. See details for description.

`defer` An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 735.

Generates a new offspring individual from the given parental genomes with the specified crossover breakpoints, queues it for addition to the target subpopulation, and returns it. The new offspring will not be visible as a member of the target subpopulation until the end of the offspring generation tick cycle stage. The target subpopulation will be used to locate applicable mutation() and modifyChild() callbacks governing the generation of the offspring individual (unlike the other addX() methods, because there are potentially up to four parental individuals to reference); recombination() callbacks will not be called by this method. This method is an advanced feature; most models will use addCrossed(), addSelfed(), or addCloned() instead. This method supports several possible configurations for strand1, strand2, and breaks1 (and the same applies for strand3, strand4, and breaks2). If strand1 and strand2 are both NULL, the corresponding genome in the generated offspring will be empty, as from addEmpty(), with no parental genomes and no added mutations; in this case, breaks1 must be NULL or zero-length. If strand1 is non-NULL but strand2 is NULL, the corresponding genome in the generated offspring will be a clonal copy of strand1 with mutations added, as from addCloned(); in this case, breaks1 must similarly be NULL or zero-length. If strand1 and strand2 are both non-NULL, the corresponding genome in the generated offspring will result from recombination between strand1 and strand2 with mutations added, as from addCrossed(), with strand1 being the initial copy strand; copying will switch between strands at each breakpoint in breaks1, which must be non-NULL but need not be sorted or uniqued (SLiM will sort and unique the supplied breakpoints internally). (It is not currently legal for strand1 to be NULL and strand2 non-NULL; that variant may be assigned some meaning in future.) Again, this discussion applies equally to strand3, strand4, and breaks2, mutatis mutandis. Note that when new mutations are generated by addRecombinant(), their subpopID property will be the id of the offspring's subpopulation, since the parental subpopulation is ambiguous in the general case; this behavior differs from the other add...() methods. The sex parameter is interpreted exactly as in addCrossed(); see that method for discussion. If the offspring sex is specified

in any way (i.e., if sex is non-NULL), the strands provided must be compatible with the sex chosen. If the offspring sex is not specified (i.e., if sex is NULL), the sex will be inferred from the strands provided where possible (when modeling an X or Y chromosome), or will be chosen randomly otherwise (when modeling autosomes); it will not be inferred from the sex of the individuals possessing the parental strands, even when the reproductive mode is essentially clonal from a single parent, since such inference would be ambiguous in the general case. When modeling the X or Y, strand1 and strand2 must be X genomes (or NULL), and strand3 and strand4 must both be X genomes or both be Y genomes (or NULL). By default, the offspring is considered to have no parents for the purposes of pedigree tracking, since there may be more than two "parents" in the general case. If pedigree tracking of parentage is desired, parent1 and/or parent2 may be passed to explicitly establish particular individuals as the parents of the offspring for purposes of pedigree tracking. In this case, if only one of parent1 and parent2 is non-NULL, that individual will be set as both of the parents of the offspring, mirroring the way that parentage is tracked for other cases such as addCloned() and addSelfed(). It is not required for parent1 or parent2 to actually be a genetic parent of the offspring at all, although typically they would be. If randomizeStrands is F (the default), strand1 will be the initial copy strand when generating the first gamete to form the offspring, and strand3 will be the initial copy strand when generating the second gamete. If randomizeStrands is T, then if strand1 and strand2 are both non-NULL, 50 will be swapped, making strand2 the initial copy strand for the first gamete; and similarly, if strand3 and strand4 are both non-NULL, 50 will be swapped, making strand4 the initial copy strand for the second gamete. This is probably usually the desired behavior, to avoid an inheritance bias due to a lack of randomization in the initial copy strand, so passing T for randomizeStrands is recommended unless you specifically desire otherwise. It is not the default behavior only for reasons of backward compatibility. These semantics allow several uses for addRecombinant(). When all strands are non-NULL, it is similar to addCrossed() except that the recombination breakpoints are specified explicitly, allowing very precise offspring generation without having to override SLiM's breakpoint generation with a recombination() callback. When only strand1 and strand3 are supplied, it is very similar to addCloned(), creating a clonal offspring, except that the two parental genomes need not belong to the same individual (whatever that might mean biologically). Supplying only strand1 is useful for modeling clonally reproducing haploids; the second genome of every offspring will be kept empty and will not receive new mutations. For a model of clonally reproducing haploids that undergo horizontal gene transfer (HGT), supplying only strand1 and strand2 will allow HGT from strand2 to replace segments of an otherwise clonal copy of strand1, while the second genome of the generated offspring will again be kept empty; this could be useful for modeling bacterial conjugation, for example. Other variations are also possible. The value of the meanParentAge property of the generated offspring is calculated from the mean parent age of each of its two genomes (whether they turn out to be null genomes or not); that may be an average of two values (if both offspring genomes have at least one parent), a single value (if one offspring genome has no parent), or no values (if both offspring genomes have no parent, in which case 0.0 results). The mean parent age of a given offspring genome is the mean of the ages of the parents of the two strands used to generate that offspring genome; if one strand is NULL then the mean parent age for that offspring genome is the age of the parent of the non-NULL strand, while if both strands are NULL then that offspring genome is parentless and is not used in the final calculation. In other words, if one offspring genome has two parents with ages A and B, and the other offspring genome has one parent with age C, the meanParentAge

of the offspring will be (A+B+C+C) / 4, not (A+B+C) / 3. Note that gene conversion tracts are not explicitly supported by this method; the breaks vectors provide crossover breakpoints, which may be used to implement crossovers or simple gene conversion tracts. There is no way to specify complex gene conversion tracts with heteroduplex mismatch repair. Beginning in SLiM 4.1, the count parameter dictates how many offspring will be generated (previously, exactly one offspring was generated). Each offspring is generated independently, based upon the given parameters. The returned vector contains all generated offspring, except those that were rejected by a modifyChild() callback. If all offspring are rejected, object<Individual>(0) is returned, which is a zero-length object vector of class Individual; note that this is a change in behavior from earlier versions, which would return NULL. Beginning in SLiM 4.1, passing T for defer will defer the generation of the genomes of the produced offspring until the end of the reproduction phase. Genome generation can only be deferred if there are no active mutation() callbacks; otherwise, an error will result. Furthermore, when genome generation is deferred the mutations of the genomes of the generated offspring may not be accessed until reproduction is complete (whether from a modifyChild() callback or otherwise). There is little or no advantage to deferring genome generation at this time (it is in place for future expansion); the default of F for defer is generally preferable since it has fewer restrictions. Also beginning in SLiM 4.1, in spatial models the spatial position of the offspring will be inherited (i.e., copied) from parent1; more specifically, the x property will be inherited in all spatial models (1D/2D/3D), the y property in 2D/3D models, and the z property in 3D models. Properties not inherited will be left uninitialized, as they were prior to SLiM 4.1. The parent's spatial position is probably not desirable in itself; the intention here is to make it easy to model the natal dispersal of all the new offspring for a given tick with a single vectorized call to pointDeviated(). If parent1 is NULL (the default), parent2 will be used; if it is also NULL, no spatial position will be inherited. Note that this method is only for use in nonWF models. See addCrossed() for further general notes on the addition of new offspring individuals.

Value

An object of type Individual object.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Subpopulation: P, addCloned(), addCrossed(), addEmpty(), addSelfed(), addSpatialMap(), cachedFitness(), configureDisplay(), defineSpatialMap(), outputMSSample(), outputSample(), outputVCFSample(), pointDeviated(), pointInBounds(), pointPeriodic(), pointReflected(), pointStopped(), pointUniform(), removeSpatialMap(), removeSubpopulation(), sampleIndividuals(),

setCloningRate(), setMigrationRates(), setSelfingRate(), setSexRatio(), setSpatialBounds(), setSubpopulationSize(), spatialMapColor(), spatialMapImage(), spatialMapValue(), subsetIndividuals(), takeMigrants()

`addSelfed` *SLiM method addSelfed*

Description

Documentation for SLiM function `addSelfed`, which is a method of the SLiM class `Subpopulation`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
addSelfed(parent, count, defer)
```

Arguments

`parent`	An object of type Individual object. Must be of length 1 (a singleton). See details for description.
`count`	An object of type integer. Must be of length 1 (a singleton). The default value is `1`. See details for description.
`defer`	An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 737.

Generates a new offspring individual from the given parent by selfing, queues it for addition to the target subpopulation, and returns it. The new offspring will not be visible as a member of the target subpopulation until the end of the offspring generation tick cycle stage. The subpopulation of parent will be used to locate applicable mutation(), recombination(), and modifyChild() callbacks governing the generation of the offspring individual. Since selfing requires that parent act as a source of both a male and a female gamete, this method may be called only in hermaphroditic models; calling it in sexual models will result in an error. This method represents a non-incidental selfing event, so the preventIncidentalSelfing flag of initializeSLiMOptions() has no effect on this method (in contrast to the behavior of addCrossed(), where selfing is assumed to be incidental). Beginning in SLiM 4.1, the count parameter dictates how many offspring will be generated (previously, exactly one offspring was generated). Each offspring is generated independently, based upon the given parameters. The returned vector contains all generated offspring, except those that were rejected by a modifyChild() callback. If all offspring are rejected, object<Individual>(0) is returned, which is a zero-length object vector of class Individual; note that this is a change in behavior from earlier versions, which would return NULL. Beginning in SLiM 4.1, passing

T for defer will defer the generation of the genomes of the produced offspring until the end of the reproduction phase. Genome generation can only be deferred if there are no active mutation() or recombination() callbacks; otherwise, an error will result. Furthermore, when genome generation is deferred the mutations of the genomes of the generated offspring may not be accessed until reproduction is complete (whether from a modifyChild() callback or otherwise). There is little or no advantage to deferring genome generation at this time (it is in place for future expansion); the default of F for defer is generally preferable since it has fewer restrictions. Also beginning in SLiM 4.1, in spatial models the spatial position of the offspring will be inherited (i.e., copied) from parent; more specifically, the x property will be inherited in all spatial models (1D/ 2D/3D), the y property in 2D/3D models, and the z property in 3D models. Properties not inherited will be left uninitialized, as they were prior to SLiM 4.1. The parent's spatial position is probably not desirable in itself; the intention here is to make it easy to model the natal dispersal of all the new offspring for a given tick with a single vectorized call to pointDeviated(). Note that this method is only for use in nonWF models. See addCrossed() for further general notes on the addition of new offspring individuals.

Value

An object of type Individual object.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Subpopulation: P, addCloned(), addCrossed(), addEmpty(), addRecombinant(), addSpatialMap(), cachedFitness(), configureDisplay(), defineSpatialMap(), outputMSSample(), outputSample(), outputVCFSample(), pointDeviated(), pointInBounds(), pointPeriodic(), pointReflected(), pointStopped(), pointUniform(), removeSpatialMap(), removeSubpopulation(), sampleIndividuals(), setCloningRate(), setMigrationRates(), setSelfingRate(), setSexRatio(), setSpatialBounds(), setSubpopulationSize(), spatialMapColor(), spatialMapImage(), spatialMapValue(), subsetIndividuals(), takeMigrants()

`addSpatialMap` *SLiM method addSpatialMap*

Description

Documentation for SLiM function `addSpatialMap`, which is a method of the SLiM class `Subpopulation`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
addSpatialMap(map)
```

Arguments

`map`	An object of type SpatialMap object. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 737.

Adds the given SpatialMap object, map, to the subpopulation. (The spatial map would have been previously created with a call to defineSpatialMap() on a different subpopulation; addSpatialMap() can then be used to add that existing spatial map with other subpopulations, sharing the map between subpopulations.) If the map is already added to the target subpopulation, this method does nothing; if a different map with the same name is already added to the subpopulation, an error results (because map names must be unique within each subpopulation). The map being added must be compatible with the target subpopulation; in particular, the spatial bounds utilized by the map must exactly match the corresponding spatial bounds for the subpopulation, and the dimensionality of the subpopulation must encompass the spatiality of the map. For example, if the map has a spatiality of "xz" then the subpopulation must have a dimensionality of "xyz" so that it encompasses both "x" and "z", and the subpopulation's spatial bounds for "x" and "z" must match those for the map (but the spatial bounds for "y" are unimportant, since the map does not use that dimension). Adding a map to a subpopulation is not strictly necessary, at present; one may query a SpatialMap object directly using mapValue(), regarding points in a subpopulation, without the map actually having been added to that subpopulation. However, it is a good idea to use addSpatialMap(), both for its compatibility check that prevents unnoticed scripting errors, and because it ensures correct display of the model in SLiMgui.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Subpopulation: P, addCloned(), addCrossed(), addEmpty(), addRecombinant(), addSelfed(), cachedFitness(), configureDisplay(), defineSpatialMap(), outputMSSample(), outputSample(), outputVCFSample(), pointDeviated(), pointInBounds(), pointPeriodic(), pointReflected(), pointStopped(), pointUniform(), removeSpatialMap(), removeSubpopulation(), sampleIndividuals(), setCloningRate(), setMigrationRates(), setSelfingRate(), setSexRatio(), setSpatialBounds(), setSubpopulationSize(), spatialMapColor(), spatialMapImage(), spatialMapValue(), subsetIndividuals(), takeMigrants()

---

addSubpop *SLiM method addSubpop*

Description

Documentation for SLiM function `addSubpop`, which is a method of the SLiM class Species. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
addSubpop(subpopID, size, sexRatio, haploid)
```

Arguments

`subpopID`	An object of type integer or string. Must be of length 1 (a singleton). See details for description.
`size`	An object of type integer. Must be of length 1 (a singleton). See details for description.
`sexRatio`	An object of type float. Must be of length 1 (a singleton). The default value is `0.5`. See details for description.
`haploid`	An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 719.

Add a new subpopulation with id subpopID and size individuals. The subpopID parameter may be either an integer giving the ID of the new subpopulation, or a string giving the name of the new subpopulation (such as "p5" to specify an ID of 5). Only if sex is enabled for the species, the initial sex ratio may optionally be specified as sexRatio (as the male fraction, M:M+F); if it is not specified, a default of 0.5 is used. The new subpopulation will be defined as a global variable immediately by this method (see section 25.16), and will also be returned by this method. Subpopulations added by this method will initially consist of individuals with empty genomes. In order to model subpopulations that split from an already existing subpopulation, use addSubpopSplit(). Only in nonWF models, the haploid parameter may be T; in this case, the second genome of each new individual will be a null genome, rather than an empty genome. For even greater control in nonWF models, you can call addSubpop() with an initial size of 0 and then stock the population with new individuals created however you wish in the next tick's reproduction() callback.

Value

An object of type Subpopulation object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Species: Sp, addSubpopSplit(), countOfMutationsOfType(), individualsWithPedigreeIDs(), killIndividuals(), mutationCounts(), mutationFrequencies(), mutationsOfType(), outputFixedMutations(), outputFull(), outputMutations(), readFromPopulationFile(), recalculateFitness(), registerFitnessEffectCallback(), registerMateChoiceCallback(), registerModifyChildCallback(), registerMutationCallback(), registerMutationEffectCallback(), registerRecombinationCallback(), registerReproductionCallback(), registerSurvivalCallback(), simulationFinished(), skipTick(), subsetMutations(), treeSeqCoalesced(), treeSeqOutput(), treeSeqRememberIndividuals(), treeSeqSimplify()

`addSubpopSplit` *SLiM method addSubpopSplit*

Description

Documentation for SLiM function `addSubpopSplit`, which is a method of the SLiM class `Species`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
addSubpopSplit(subpopID, size, sourceSubpop, sexRatio)
```

Arguments

`subpopID`	An object of type integer or string. Must be of length 1 (a singleton). See details for description.
`size`	An object of type integer. Must be of length 1 (a singleton). See details for description.
`sourceSubpop`	An object of type integer or Subpopulation object. Must be of length 1 (a singleton). See details for description.
`sexRatio`	An object of type float. Must be of length 1 (a singleton). The default value is `0.5`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 720.

Split off a new subpopulation with id subpopID and size individuals derived from subpopulation sourceSubpop. The subpopID parameter may be either an integer giving the ID of the new subpopulation, or a string giving the name of the new subpopulation (such as "p5" to specify an ID of 5). The sourceSubpop parameter may specify the source subpopulation either as a Subpopulation object or by integer identifier. Only if sex is enabled for the species, the initial sex ratio may optionally be specified as sexRatio (as the male fraction, M:M+F); if it is not specified, a default of 0.5 is used. The new subpopulation will be defined as a global variable immediately by this method (see section 25.16), and will also be returned by this method. Subpopulations added by this method will consist of individuals that are clonal copies of individuals from the source subpopulation, randomly chosen with probabilities proportional to fitness. The fitness of all of these initial individuals is considered to be 1.0, to avoid a doubled round of selection in the initial tick, given that fitness values were already used to choose the individuals to clone. Once this initial set of individuals has mated to produce offspring, the model is effectively of parental individuals in the source subpopulation mating randomly according to fitness, as usual in SLiM, with juveniles migrating to the newly added subpopulation. Effectively, then, then new subpopulation is created empty, and is filled by migrating juveniles from the source subpopulation, in accordance with SLiM's usual model of juvenile migration.

Value

An object of type Subpopulation object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Species: Sp, addSubpop(), countOfMutationsOfType(), individualsWithPedigreeIDs(), killIndividuals(), mutationCounts(), mutationFrequencies(), mutationsOfType(), outputFixedMutations(), outputFull(), outputMutations(), readFromPopulationFile(), recalculateFitness(), registerFitnessEffectCallback(), registerMateChoiceCallback(), registerModifyChildCallback(), registerMutationCallback(), registerMutationEffectCallback(), registerRecombinationCallback(), registerReproductionCallback(), registerSurvivalCallback(), simulationFinished(), skipTick(), subsetMutations(), treeSeqCoalesced(), treeSeqOutput(), treeSeqRememberIndividuals(), treeSeqSimplify()

---

addSubpopulationSexRatio

SLiM method addSubpopulationSexRatio

Description

Documentation for SLiM function addSubpopulationSexRatio, which is a method of the SLiM class LogFile. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
addSubpopulationSexRatio(subpop)
```

Arguments

subpop	An object of type integer or Subpopulation object. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 702.

Adds a new data column that provides the sex ratio M:(M+F) of the subpopulation subpop, named pX_sex_ratio. If the subpopulation exists but has a size of zero, NA will be written.

Value

An object of type void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or logical.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other LogFile: LF, addCustomColumn(), addCycleStage(), addCycle(), addKeysAndValuesFrom(), addMeanSDColumns(), addPopulationSexRatio(), addPopulationSize(), addSubpopulationSize(), addSuppliedColumn(), addTick(), clearKeysAndValues(), flush(), logRow(), setFilePath(), setLogInterval(), setSuppliedValue(), setValue(), willAutolog()

---

addSubpopulationSize *SLiM method addSubpopulationSize*

Description

Documentation for SLiM function addSubpopulationSize, which is a method of the SLiM class LogFile. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
addSubpopulationSize(subpop)
```

Arguments

subpop	An object of type integer or Subpopulation object. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 702.

Adds a new data column that provides the size of the subpopulation subpop, named pX_num_individuals. If the subpopulation exists but has a size of zero, 0 will be written.

Value

An object of type void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or logical.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other LogFile: LF, addCustomColumn(), addCycleStage(), addCycle(), addKeysAndValuesFrom(), addMeanSDColumns(), addPopulationSexRatio(), addPopulationSize(), addSubpopulationSexRatio(), addSuppliedColumn(), addTick(), clearKeysAndValues(), flush(), logRow(), setFilePath(), setLogInterval(), setSuppliedValue(), setValue(), willAutolog()

---

addSuppliedColumn *SLiM method addSuppliedColumn*

Description

Documentation for SLiM function addSuppliedColumn, which is a method of the SLiM class LogFile. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
addSuppliedColumn(columnName)
```

Arguments

columnName	An object of type string. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 702.

Adds a new data column with its name provided by columnName. The value for the column is initially undefined, and will be written as NA. A different value may (optionally) be provided by calling setSuppliedValue() with a value for columnName. That value will be used for the column the next time a row is generated (whether automatically or by a call to logRow()), and the column's value will subsequently be undefined again. In other words, for any given logged row the default of NA may be kept, or a different value may be supplied. This allows the value for the column to be set at any point during the tick cycle, which can be convenient if the column's value depends upon transient state that is no longer available at the time the row is logged.

Value

An object of type void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or logical.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other LogFile: LF, addCustomColumn(), addCycleStage(), addCycle(), addKeysAndValuesFrom(), addMeanSDColumns(), addPopulationSexRatio(), addPopulationSize(), addSubpopulationSexRatio(), addSubpopulationSize(), addTick(), clearKeysAndValues(), flush(), logRow(), setFilePath(), setLogInterval(), setSuppliedValue(), setValue(), willAutolog()

---

addTick *SLiM method addTick*

Description

Documentation for SLiM function addTick, which is a method of the SLiM class LogFile. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
addTick(void)
```

Arguments

`void`	An object of type . See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 702.

Adds a new data column, named tick, that provides the tick number for the simulation.

Value

An object of type void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or logical.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other LogFile: `LF`, `addCustomColumn()`, `addCycleStage()`, `addCycle()`, `addKeysAndValuesFrom()`, `addMeanSDColumns()`, `addPopulationSexRatio()`, `addPopulationSize()`, `addSubpopulationSexRatio()`, `addSubpopulationSize()`, `addSuppliedColumn()`, `clearKeysAndValues()`, `flush()`, `logRow()`, `setFilePath()`, `setLogInterval()`, `setSuppliedValue()`, `setValue()`, `willAutolog()`

`ancestralNucleotides` *SLiM method ancestralNucleotides*

Description

Documentation for SLiM function `ancestralNucleotides`, which is a method of the SLiM class `Chromosome`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
ancestralNucleotides(start, end, format = "string")
```

Arguments

`start`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`end`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`format`	= "string" An object of type string. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 661.

Returns the ancestral nucleotide sequence originally supplied to initializeAncestralNucleotides(), including any sequence changes due to nucleotide mutations that have fixed and substituted. This nucleotide sequence is the reference sequence for positions in a genome that do not contain a nucleotide-based mutation. The range of the returned sequence may be constrained by a start position given in start and/or an end position given in end; nucleotides will be returned from start to end, inclusive. The default value of NULL for start and end represent the first and last base positions of the chromosome, respectively. The format of the returned sequence is controlled by the format parameter. A format of "string" will return the sequence as a singleton string (e.g., "TATA"). A format of "char" will return a string vector with one element per nucleotide (e.g., "T", "A", "T", "A"). A format of "integer" will return an integer vector with values A=0, C=1, G=2, T=3 (e.g., 3, 0, 3, 0). If the sequence returned is likely to be long, the "string" format will be the most memory-efficient, and may also be the fastest (but may be harder to work with). For purposes related to interpreting the nucleotide sequence as a coding sequence, a format of "codon" is also supported. This format will return an integer vector with values from 0 to 63, based upon successive nucleotide triplets in the sequence (which, for this format, must have a length that is a multiple of three). The codon value for a given nucleotide triplet XYZ is 16X + 4Y + Z, where X, Y, and Z have the usual values A=0, C=1, G=2, T=3. For example, the triplet AAA has a codon value of 0, AAC is 1, AAG is 2, AAT is 3, ACA is 4, and on upward to TTT which is 63. If the nucleotide sequence AACACATTT is requested in codon format, the codon vector 1 4 63 will therefore be returned. These codon values can be useful in themselves; they can also be passed to codonsToAminoAcids() to translate them into the corresponding amino acid sequence if desired (see section 25.18.1).

Value

An object of type integer or string.

Copyright

More information about SLiM can be found on the official website: https://messerlab.org/slim/

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Chromosome: Ch, drawBreakpoints(), setAncestralNucleotides(), setGeneConversion(), setHotspotMap(), setMutationRate(), setRecombinationRate()

`as_slimr_code` *Convert SLiM code text into equivalent 'slimr' code to produce the same model*

Description

This function print the equivalent 'slimr' code to the console where it can easily be copied and pasted into an R script.

Usage

```
as_slimr_code(code_txt)
```

Arguments

`code_txt`

Value

'slimr' code as a character vector, invisibly

Examples

```
as_slimr_code(slim_recipes$`5.3.4`)
```

as_slimr_script *Convert a character vector into a slim_script object*

Description

Convert a character vector into a slim_script object

Usage

```
as_slimr_script(slim_script_text)
```

Arguments

slim_script_text	A character vector giving the full SLiM script to convert to a `slimr_script` object. Character vectors with length greater than 1 will be concatenated with newline separators

Value

A `slimr_script` object

Examples

```
cat(slim_recipes[[1]])
as_slimr_script(slim_recipes[[1]])
```

as_slim_text *Convert a slimr_script to a length 1 character vector*

Description

Convert a slimr_script to a length 1 character vector

Usage

```
as_slim_text(x, ...)
```

Arguments

x	`slimr_script` object to convert
...	Further arguments, passed to or from other methods.

Value

A length 1 character vector

Examples

```
slim_script(
   slim_block(initialize(),
          {
            initializeMutationRate(1e-7);
            initializeMutationType("m1", 0.5, "f", 0.0);
            initializeGenomicElementType("g1", m1, 1.0);
            initializeGenomicElement(g1, 0, 99999);
            initializeRecombinationRate(1e-8);
          }),
   slim_block(1,
          {
            sim.addSubpop("p1", 500);
          }),
   slim_block(10000,
          {
            sim.simulationFinished();
          })
) -> script
as_slim_text(script)
```

---

as_slim_text.slimr_script

Convert a slimr_script to a length 1 character vector

Description

Convert a slimr_script to a length 1 character vector

Usage

```
## S3 method for class 'slimr_script'
as_slim_text(x, ...)
```

Arguments

x	`slimr_script` object to convert
...	Further arguments, passed to or from other methods.

Value

A length 1 character vector

Examples

```
slim_script(
   slim_block(initialize(),
          {
            initializeMutationRate(1e-7);
```

```
            initializeMutationType("m1", 0.5, "f", 0.0);
            initializeGenomicElementType("g1", m1, 1.0);
            initializeGenomicElement(g1, 0, 99999);
            initializeRecombinationRate(1e-8);
          }),
    slim_block(1,
          {
            sim.addSubpop("p1", 500);
          }),
    slim_block(10000,
          {
            sim.simulationFinished();
          })
) -> script
as_slim_text(script)
```

`blend` *SLiM method blend*

Description

Documentation for SLiM function `blend`, which is a method of the SLiM class `SpatialMap`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
blend(x, xFraction)
```

Arguments

`x`	An object of type integer or float or SpatialMap object. See details for description.
`xFraction`	An object of type float. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 713.

Blends x into the spatial map, giving x a weight of xFraction and the existing values in the target spatial map a weight of 1 - xFraction, such that the resulting values in the target spatial map are then given by x * xFraction + target * (1 - xFraction). The value of xFraction must be in [0.0, 1.0]. One possibility is that x is a singleton integer or float value; in this case, x is blended with each grid value of the target spatial map. Another possibility is that x is an integer or float vector/ matrix/array of the same dimensions as the target spatial map's grid; in this case, each value of x is blended with the corresponding

grid value of the target spatial map. The third possibility is that x is itself a (singleton) spatial map; in this case, each grid value of x is blended with the corresponding grid value of the target spatial map (and thus the two spatial maps must match in their spatiality, their spatial bounds, and their grid dimensions). The target spatial map is returned, to allow easy chaining of operations.

Value

An object of type SpatialMap object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other SpatialMap: SM, add(), changeColors(), changeValues(), divide(), exp(), gridValues(), interpolate(), mapColor(), mapImage(), mapValue(), multiply(), power(), range(), rescale(), sampleImprovedNearbyPoint(), sampleNearbyPoint(), smooth(), subtract()

---

cachedFitness *SLiM method cachedFitness*

Description

Documentation for SLiM function `cachedFitness`, which is a method of the SLiM class `Subpopulation`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
cachedFitness(indices)
```

Arguments

`indices`	An object of type null or integer. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 738.

The fitness values calculated for the individuals at the indices given are returned. If NULL is passed, fitness values for all individuals in the subpopulation are returned. The fitness values returned are cached values; mutationEffect() and fitnessEffect() callbacks are therefore not called as a side effect of this method. It is always an error to call cachedFitness() from inside a mutationEffect() or fitnessEffect() callback, since fitness values are in the middle of being set up. In WF models, it is also an error to call cachedFitness() from a late() event, because fitness values for the new offspring generation have not yet been calculated and are undefined. In nonWF models, the population may be a mixture of new and old individuals, so instead, NAN will be returned as the fitness of any new individuals whose fitness has not yet been calculated. When new subpopulations are first created with addSubpop() or addSubpopSplit(), the fitness of all of the newly created individuals is considered to be 1.0 until fitness values are recalculated.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Subpopulation: P, addCloned(), addCrossed(), addEmpty(), addRecombinant(), addSelfed(), addSpatialMap(), configureDisplay(), defineSpatialMap(), outputMSSample(), outputSample(), outputVCFSample(), pointDeviated(), pointInBounds(), pointPeriodic(), pointReflected(), pointStopped(), pointUniform(), removeSpatialMap(), removeSubpopulation(), sampleIndividuals(), setCloningRate(), setMigrationRates(), setSelfingRate(), setSexRatio(), setSpatialBounds(), setSubpopulationSize(), spatialMapColor(), spatialMapImage(), spatialMapValue(), subsetIndividuals(), takeMigrants()

`calcFST` *SLiM method calcFST*

Description

Documentation for SLiM function `calcFST`, which is a method of the SLiM class `SLiMBuiltin`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
calcFST(genomes1, genomes2, muts, start, end)
```

Arguments

`genomes1`	An object of type Genome object. See details for description.
`genomes2`	An object of type Genome object. See details for description.
`muts`	An object of type null or Mutation object. The default value is `NULL`. See details for description.
`start`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`end`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 751.

Calculates the FST between two Genome vectors - typically, but not necessarily, the genomes that constitute two different subpopulations (which we will assume for the purposes of this discussion). In general, higher FST indicates greater genetic divergence between subpopulations. The calculation is done using only the mutations in muts; if muts is NULL, all mutations are used. The muts parameter can therefore be used to calculate the FST only for a particular mutation type (by passing only mutations of that type). The calculation can be narrowed to apply to only a window - a subrange of the full chromosome - by passing the interval bounds [start, end] for the desired window. In this case, the vector of mutations used for the calculation will be subset to include only mutations within the specified window. The default behavior, with start and end of NULL, provides the genome-wide FST, which is often used to assess the overall level of genetic divergence between sister species or allopatric subpopulations. The code for calcFST() is, roughly, an Eidos implementation of Wright's definition of FST (but see below for further discussion and clarification): where HS is the average heterozygosity in the two subpopulations, and HT is the total heterozygosity when both subpopulations are combined. In this implementation, the two genome vectors are weighted equally, not weighted by their size. In SLiM 3, the implementation followed Wright's definition closely, and returned the average of ratios: mean(1.0 - H_s/H_t), in the Eidos code. In SLiM 4, it returns the ratio of averages instead: 1.0 - mean(H_s)/mean(H_t). In other words, the FST value reported by SLiM 4 is an average across the specified mutations in the two sets of genomes, where H_s and H_t are first averaged across all specified mutations prior to taking the ratio of the two. This ratio of averages is less biased than the average of ratios, and and is generally considered to be best

practice (see, e.g., Bhatia et al., 2013). This means that the behavior of calcFST() differs between SLiM 3 and SLiM 4. The implementation of calcFST(), viewable with functionSource(), treats every mutation in muts as independent in the heterozygosity calculations; in other words, if mutations are stacked, the heterozygosity calculated is by mutation, not by site. Similarly, if multiple Mutation objects exist in different genomes at the same site (whether representing different genetic states, or multiple mutational lineages for the same genetic state), each Mutation object is treated separately for purposes of the heterozygosity calculation, just as if they were at different sites. One could regard these choices as embodying an infinite-sites interpretation of the segregating mutations. In most biologically realistic models, such genetic states will be quite rare, and so the impact of these choices will be negligible; however, in some models these distinctions may be important.

Value

An object of type float. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other SLiMBuiltin: SB, calcHeterozygosity(), calcInbreedingLoad(), calcPairHeterozygosity(), calcVA(), calcWattersonsTheta(), codonsToAminoAcids(), mm16To256(), mmJukesCantor(), mmKimura(), nucleotideCounts(), nucleotideFrequencies(), nucleotidesToCodons(), summarizeIndividuals(), treeSeqMetadata()

Examples

```
## This just brings up the documentation:
calcFST()
```

---

calcHeterozygosity *SLiM method calcHeterozygosity*

Description

Documentation for SLiM function `calcHeterozygosity`, which is a method of the SLiM class `SLiMBuiltin`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
calcHeterozygosity(genomes, muts, start, end)
```

Arguments

`genomes`	An object of type Genome object. See details for description.
`muts`	An object of type null or Mutation object. The default value is `NULL`. See details for description.
`start`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`end`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 751.

Calculates the heterozygosity for a vector of genomes, based upon the frequencies of mutations in the genomes. The result is the expected heterozygosity, for the individuals to which the genomes belong, assuming that they are under Hardy-Weinberg equilibrium; this can be compared to the observed heterozygosity of an individual, as calculated by calcPairHeterozygosity(). Often genomes will be all of the genomes in a subpopulation, or in the entire population, but any genome vector may be used. By default, with muts=NULL, the calculation is based upon all mutations in the simulation; the calculation can instead be based upon a subset of mutations, such as mutations of a specific mutation type, by passing the desired vector of mutations for muts. FST = 1 − HS HT The calculation can be narrowed to apply to only a window - a subrange of the full chromosome - by passing the interval bounds [start, end] for the desired window. In this case, the vector of mutations used for the calculation will be subset to include only mutations within the specified window. The default behavior, with start and end of NULL, provides the genome-wide heterozygosity. The implementation of calcHeterozygosity(), viewable with functionSource(), treats every mutation as independent in the heterozygosity calculations. One could regard this choice as embodying an infinite-sites interpretation of the segregating mutations. In most biologically realistic models, such genetic states will be quite rare, and so the impact of this choice will be negligible; however, in some models this distinction may be important. See calcPairHeterozygosity() for further discussion.

Value

An object of type float. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other SLiMBuiltin: SB, calcFST(), calcInbreedingLoad(), calcPairHeterozygosity(), calcVA(), calcWattersonsTheta(), codonsToAminoAcids(), mm16To256(), mmJukesCantor(), mmKimura(), nucleotideCounts(), nucleotideFrequencies(), nucleotidesToCodons(), summarizeIndividuals(), treeSeqMetadata()

Examples

```
## This just brings up the documentation:
calcHeterozygosity()
```

---

calcInbreedingLoad *SLiM method calcInbreedingLoad*

Description

Documentation for SLiM function calcInbreedingLoad, which is a method of the SLiM class SLiMBuiltin. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
calcInbreedingLoad(genomes, mutType)
```

Arguments

genomes	An object of type Genome object. See details for description.
mutType	An object of type null or MutationType object. Must be of length 1 (a singleton). The default value is NULL. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 752.

Calculates inbreeding load (the haploid number of lethal equivalents, or B) for a vector of genomes passed in genomes. The calculation can be limited to a focal mutation type passed in mutType; if mutType is NULL (the default), all of the mutations for the focal species will be considered. In any case, only deleterious mutations (those with a negative selection coefficient) will be included in the final calculation. The inbreeding load is a measure of the quantity of recessive deleterious variation that is heterozygous in a population and can contribute to fitness declines under inbreeding. This function implements the following equation from Morton et al. (1956), which assumes no epistasis and random mating: B =

sum(qs) − sum(q2s) − 2sum(q(1−q)sh) where q is the frequency of a given deleterious allele, s is the absolute value of the selection coefficient, and h is its dominance coefficient. Note that the implementation sets a maximum |s| of 1.0 (i.e., a lethal allele); |s| can sometimes be greater than 1.0 when s is drawn from a distribution, but in practice an allele with s < -1.0 has the same lethal effect as when s = -1.0. Also note that this implementation will not work when the model changes the dominance coefficients of mutations using mutationEffect() callbacks, since it relies on the dominanceCoeff property of MutationType. Finally, note that, to estimate the diploid number of lethal equivalents (2B), the result from this function can simply be multiplied by two. This function was contributed by Chris Kyriazis; thanks, Chris!

Value

An object of type float. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other SLiMBuiltin: SB, calcFST(), calcHeterozygosity(), calcPairHeterozygosity(), calcVA(), calcWattersonsTheta(), codonsToAminoAcids(), mm16To256(), mmJukesCantor(), mmKimura(), nucleotideCounts(), nucleotideFrequencies(), nucleotidesToCodons(), summarizeIndividuals(), treeSeqMetadata()

Examples

```
## This just brings up the documentation:
calcInbreedingLoad()
```

---

calcPairHeterozygosity

SLiM method calcPairHeterozygosity

Description

Documentation for SLiM function `calcPairHeterozygosity`, which is a method of the SLiM class `SLiMBuiltin`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
calcPairHeterozygosity(genome1, genome2, start, end, infiniteSites)
```

Arguments

`genome1`	An object of type Genome object. Must be of length 1 (a singleton). See details for description.
`genome2`	An object of type Genome object. Must be of length 1 (a singleton). See details for description.
`start`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`end`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`infiniteSites`	An object of type logical. Must be of length 1 (a singleton). The default value is `T`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 752.

Calculates the heterozygosity for a pair of genomes; these will typically be the two genomes of a diploid individual (individual.genome1 and individual.genome2), but any two genomes may be supplied. The calculation can be narrowed to apply to only a window - a subrange of the full chromosome - by passing the interval bounds [start, end] for the desired window. In this case, the vector of mutations used for the calculation will be subset to include only mutations within the specified window. The default behavior, with start and end of NULL, provides the genome-wide heterozygosity. The implementation calcPairHeterozygosity(), viewable with functionSource(), treats every mutation as independent in the heterozygosity calculations by default (i.e., with infiniteSites=T). If mutations are stacked, the heterozygosity calculated therefore depends upon the number of unshared mutations, not the number of differing sites. Similarly, if multiple Mutation objects exist in different genomes at the same site (whether representing different genetic states, or multiple mutational lineages for the same genetic state), each Mutation object is treated separately for purposes of the heterozygosity calculation, just as if they were at different sites. One could regard these choices as embodying an infinite-sites interpretation of the segregating mutations. In most biologically realistic models, such genetic states will be quite rare, and so the impact of this choice will be negligible; however, in some models this distinction may be important. The behavior of calcPairHeterozygosity() can be switched to calculate based upon the number of differing sites, rather than the number of unshared mutations, by passing infiniteSites=F.

Value

An object of type float. Return will be of length 1 (a singleton)

Copyright

More information about SLiM can be found on the official website: https://messerlab.org/slim/

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other SLiMBuiltin: SB, calcFST(), calcHeterozygosity(), calcInbreedingLoad(), calcVA(), calcWattersonsTheta(), codonsToAminoAcids(), mm16To256(), mmJukesCantor(), mmKimura(), nucleotideCounts(), nucleotideFrequencies(), nucleotidesToCodons(), summarizeIndividuals(), treeSeqMetadata()

Examples

```
## This just brings up the documentation:
calcPairHeterozygosity()
```

---

calcVA *SLiM method calcVA*

Description

Documentation for SLiM function calcVA, which is a method of the SLiM class SLiMBuiltin. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
calcVA(individuals, mutType)
```

Arguments

individuals	An object of type Individual object. See details for description.
mutType	An object of type integer or MutationType object. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 753.

Calculates VA, the additive genetic variance, among a vector individuals, in a particular mutation type mutType that represents quantitative trait loci (QTLs) influencing a quantitative phenotypic trait. The mutType parameter may be either an integer representing the ID of the desired mutation type, or a MutationType object specified directly. This

function assumes that mutations of type mutType encode their effect size upon the quantitative trait in their selectionCoeff property, as is fairly standard in SLiM (see, e.g., section 13.2). The implementation of calcVA(), which is viewable with functionSource(), is quite simple; if effect sizes are stored elsewhere (such as with setValue(), as in section 13.5), a new user-defined function following the pattern of calcVA() can easily be written.

Value

An object of type float. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other SLiMBuiltin: SB, calcFST(), calcHeterozygosity(), calcInbreedingLoad(), calcPairHeterozygosity(
calcWattersonsTheta(), codonsToAminoAcids(), mm16To256(), mmJukesCantor(), mmKimura(),
nucleotideCounts(), nucleotideFrequencies(), nucleotidesToCodons(), summarizeIndividuals(),
treeSeqMetadata()

Examples

```
## This just brings up the documentation:
calcVA()
```

---

calcWattersonsTheta *SLiM method calcWattersonsTheta*

Description

Documentation for SLiM function calcWattersonsTheta, which is a method of the SLiM class SLiMBuiltin. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
calcWattersonsTheta(genomes, muts, start, end)
```

Arguments

`genomes`	An object of type Genome object. See details for description.
`muts`	An object of type null or Mutation object. The default value is `NULL`. See details for description.
`start`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`end`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 753.

Calculates Watterson's theta (a metric of genetic diversity comparable to heterozygosity) for a vector of genomes, based upon the mutations in the genomes. Often genomes will be all of the genomes in a subpopulation, or in the entire population, but any genome vector may be used. By default, with muts=NULL, the calculation is based upon all mutations in the simulation; the calculation can instead be based upon a subset of mutations, such as mutations of a specific mutation type, by passing the desired vector of mutations for muts. The calculation can be narrowed to apply to only a window - a subrange of the full chromosome - by passing the interval bounds [start, end] for the desired window. In this case, the vector of mutations used for the calculation will be subset to include only mutations within the specified window. The default behavior, with start and end of NULL, provides the genome-wide Watterson's theta. The implementation of calcWattersonsTheta(), viewable with functionSource(), treats every mutation as independent in the heterozygosity calculations. One could regard this choice as embodying an infinite-sites interpretation of the segregating mutations, as with calcHeterozygosity(). In most biologically realistic models, such genetic states will be quite rare, and so the impact of this assumption will be negligible; however, in some models this distinction may be important. See calcPairHeterozygosity() for further discussion.

Value

An object of type float. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (`<bhaller@benhaller.com>`) and Philipp W Messer (`<messer@cornell.edu>`)

See Also

Other SLiMBuiltin: SB, calcFST(), calcHeterozygosity(), calcInbreedingLoad(), calcPairHeterozygosity(
calcVA(), codonsToAminoAcids(), mm16To256(), mmJukesCantor(), mmKimura(), nucleotideCounts(),
nucleotideFrequencies(), nucleotidesToCodons(), summarizeIndividuals(), treeSeqMetadata()

Examples

```
## This just brings up the documentation:
calcWattersonsTheta()
```

Ch *Chromosome*

Description

Documentation for Chromosome class from SLiM

Details

This class represents the layout and properties of the chromosome being simulated. The chromosome currently being simulated is available through the sim.chromosome global. Section 1.5.4 presents an overview of the conceptual role of this class. This class has the following methods (functions):

- ancestralNucleotides
- drawBreakpoints
- setAncestralNucleotides
- setGeneConversion
- setHotspotMap
- setMutationRate
- setRecombinationRate

This class has the following properties:

colorSubstitution A property of type string. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** The color used to display substitutions in SLiMgui when both mutations and substitutions are being displayed in the chromosome view. Outside of SLiMgui, this property still exists, but is not used by SLiM. Colors may be specified by name, or with hexadecimal RGB values of the form "#RRGGBB" (see the Eidos manual). If colorSubstitution is the empty string, "", SLiMgui will defer to the color scheme of each MutationType, just as it does when only substitutions are being displayed. The default, "3333FF", causes all substitutions to be shown as dark blue when displayed in conjunction with mutations, to prevent the view from becoming too noisy. Note that when substitutions are displayed without mutations also being displayed, this value is ignored by SLiMgui and the substitutions use the color scheme of each MutationType.

geneConversionEnabled A property of type logical. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** When gene conversion has been enabled by calling initializeGeneConversion(), switching to the DSB recombination model, this property is T; otherwise, when using the crossover breakpoints model, it is F.

geneConversionGCBias A property of type float. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The gene conversion bias coefficient, which expresses a bias in the resolution of heteroduplex mismatches in complex gene conversion tracts. When gene conversion has not been enabled by calling initializeGeneConversion(), this property will be unavailable.

geneConversionNonCrossoverFraction A property of type float. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The fraction of double-stranded breaks that result in non-crossover events. When gene conversion has not been enabled by calling initializeGeneConversion(), this property will be unavailable.

geneConversionMeanLength A property of type float. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The mean length of a gene conversion tract (in base positions). When gene conversion has not been enabled by calling initializeGeneConversion(), this property will be unavailable.

geneConversionSimpleConversionFraction A property of type float. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The fraction of gene conversion tracts that are "simple" (i.e., not involving resolution of heteroduplex mismatches); the remainder will be "complex". When gene conversion has not been enabled by calling initializeGeneConversion(), this property will be unavailable.

genomicElements A property of type GenomicElement object. This property is a constant, so it is not modifiable. **Property Description:** All of the GenomicElement objects that comprise the chromosome.

hotspotEndPositions A property of type integer. This property is a constant, so it is not modifiable. **Property Description:** The end positions for hotspot map regions along the chromosome. Each hotspot map region is assumed to start at the position following the end of the previous hotspot map region; in other words, the regions are assumed to be contiguous. When using sex-specific hotspot maps, this property will unavailable; see hotspotEndPositionsF and hotspotEndPositionsM.

hotspotEndPositionsF A property of type integer. This property is a constant, so it is not modifiable. **Property Description:** The end positions for hotspot map regions for females, when using sex-specific hotspot maps; unavailable otherwise. See hotspotEndPositions for further explanation.

hotspotEndPositionsM A property of type integer. This property is a constant, so it is not modifiable. **Property Description:** The end positions for hotspot map regions for males, when using sex-specific hotspot maps; unavailable otherwise. See hotspotEndPositions for further explanation.

hotspotMultipliers A property of type float. This property is a constant, so it is not modifiable. **Property Description:** The hotspot multiplier for each of the hotspot map regions specified by hotspotEndPositions. When using sex-specific hotspot maps, this property will be unavailable; see hotspotMultipliersF and hotspotMultipliersM.

hotspotMultipliersF A property of type float. This property is a constant, so it is not modifiable. **Property Description:** The hotspot multiplier for each of the hotspot map regions specified by hotspotEndPositionsF, when using sex-specific hotspot maps; unavailable otherwise.

hotspotMultipliersM A property of type float. This property is a constant, so it is not modifiable. **Property Description:** The hotspot multiplier for each of the hotspot map regions specified by hotspotEndPositionsM, when using sex-specific hotspot maps; unavailable otherwise.

lastPosition A property of type integer. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The last valid position in the chromosome; its length, essentially. Note that the chromosome length is determined by the maximum of the end of the last genomic element, the end of the last recombination region, and the end of the last mutation map region (or hotspot map region).

mutationEndPositions A property of type integer. This property is a constant, so it is not modifiable. **Property Description:** The end positions for mutation rate regions along the chromosome. Each mutation rate region is assumed to start at the position following the end of the previous mutation rate region; in other words, the regions are assumed to be contiguous. When using sex-specific mutation rate maps, this property will unavailable; see mutationEndPositionsF and mutationEndPositionsM. This property is unavailable in nucleotide-based models.

mutationEndPositionsF A property of type integer. This property is a constant, so it is not modifiable. **Property Description:** The end positions for mutation rate regions for females, when using sex-specific mutation rate maps; unavailable otherwise. See mutationEndPositions for further explanation. This property is unavailable in nucleotide-based models.

mutationEndPositionsM A property of type integer. This property is a constant, so it is not modifiable. **Property Description:** The end positions for mutation rate regions for males, when using sex-specific mutation rate maps; unavailable otherwise. See mutationEndPositions for further explanation. This property is unavailable in nucleotide-based models.

mutationRates A property of type float. This property is a constant, so it is not modifiable. **Property Description:** The mutation rate for each of the mutation rate regions specified by mutationEndPositions. When using sex-specific mutation rate maps, this property will be unavailable; see mutationRatesF and mutationRatesM. This property is unavailable in nucleotide-based models.

mutationRatesF A property of type float. This property is a constant, so it is not modifiable. **Property Description:** The mutation rate for each of the mutation rate regions specified by mutationEndPositionsF, when using sex-specific mutation rate maps; unavailable otherwise. This property is unavailable in nucleotide-based models.

mutationRatesM A property of type float. This property is a constant, so it is not modifiable. **Property Description:** The mutation rate for each of the mutation rate regions specified by mutationEndPositionsM, when using sex-specific mutation rate maps; unavailable otherwise. This property is unavailable in nucleotide-based models.

overallMutationRate A property of type float. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The overall mutation rate across the whole chromosome determining the overall number of mutation

ranges and rates as well as the coverage of the chromosome by genomic elements (since mutations are only generated within genomic elements, regardless of the mutation rate map). When using sexspecific mutation rate maps, this property will unavailable; see overallMutationRateF and overallMutationRateM. This property is unavailable in nucleotide-based models.

overallMutationRateF A property of type float. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The overall mutation rate for females, when using sex-specific mutation rate maps; unavailable otherwise. See overallMutationRate for further explanation. This property is unavailable in nucleotide-based models.

overallMutationRateM A property of type float. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The overall mutation rate for males, when using sex-specific mutation rate maps; unavailable otherwise. See overallMutationRate for further explanation. This property is unavailable in nucleotide-based models.

overallRecombinationRate A property of type float. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The overall recombination rate across the whole chromosome determining the overall number of recombination events that will occur anywhere in the chromosome, as calculated from the individual recombination ranges and rates. When using sex-specific recombination maps, this property will unavailable; see overallRecombinationRateF and overallRecombinationRateM.

overallRecombinationRateF A property of type float. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The overall recombination rate for females, when using sex-specific recombination maps; unavailable otherwise. See overallRecombinationRate for further explanation.

overallRecombinationRateM A property of type float. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The overall recombination rate for males, when using sex-specific recombination maps; unavailable otherwise. See overallRecombinationRate for further explanation.

recombinationEndPositions A property of type integer. This property is a constant, so it is not modifiable. **Property Description:** The end positions for recombination regions along the chromosome. Each recombination region is assumed to start at the position following the end of the previous recombination region; in other words, the regions are assumed to be contiguous. When using sex-specific recombination maps, this property will unavailable; see recombinationEndPositionsF and recombinationEndPositionsM.

recombinationEndPositionsF A property of type integer. This property is a constant, so it is not modifiable. **Property Description:** The end positions for recombination regions for females, when using sex-specific recombination maps; unavailable otherwise. See recombinationEndPositions for further explanation.

recombinationEndPositionsM A property of type integer. This property is a constant, so it is not modifiable. **Property Description:** The end positions for recombination regions for males, when using sex-specific recombination maps; unavailable otherwise. See recombinationEndPositions for further explanation.

recombinationRates A property of type float. This property is a constant, so it is not modifiable. **Property Description:** The recombination rate for each of the recombination regions specified by recombinationEndPositions. When using sex-specific

recombination maps, this property will unavailable; see recombinationRatesF and recombinationRatesM.

recombinationRatesF A property of type float. This property is a constant, so it is not modifiable. **Property Description:** The recombination rate for each of the recombination regions specified by recombinationEndPositionsF, when using sex-specific recombination maps; unavailable otherwise.

recombinationRatesM A property of type float. This property is a constant, so it is not modifiable. **Property Description:** The recombination rate for each of the recombination regions specified by recombinationEndPositionsM, when using sex-specific recombination maps; unavailable otherwise.

species A property of type Species object. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The species to which the target object belongs.

tag A property of type integer. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** A user-defined integer value. The value of tag is initially undefined, and it is an error to try to read it; if you wish it to have a defined value, you must arrange that yourself by explicitly setting its value prior to using it elsewhere in your code. The value of tag is not used by SLiM; it is free for you to use.

See Also

Other Chromosome: `ancestralNucleotides()`, `drawBreakpoints()`, `setAncestralNucleotides()`, `setGeneConversion()`, `setHotspotMap()`, `setMutationRate()`, `setRecombinationRate()`

`changeColors` *SLiM method changeColors*

Description

Documentation for SLiM function `changeColors`, which is a method of the SLiM class `SpatialMap`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
changeColors(valueRange, colors)
```

Arguments

`valueRange`	An object of type null or integer or float. The default value is `NULL`. See details for description.
`colors`	An object of type null or string. The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 713.

Changes the color scheme for the target spatial map. The meaning of valueRange and colors are identical to their meaning in defineSpatialMap(), but are also described here. The valueRange and colors parameters travel together; either both are NULL, or both are specified. They control how map values will be transformed into colors, by SLiMgui and by the mapColor() method. The valueRange parameter establishes the color-mapped range of spatial map values, as a vector of length two specifying a minimum and maximum; this does not need to match the actual range of values in the map. The colors parameter then establishes the corresponding colors for values within the interval defined by valueRange: values less than or equal to valueRange[0] will map to colors[0], values greater than or equal to valueRange[1] will map to the last colors value, and intermediate values will shade continuously through the specified vector of colors, with interpolation between adjacent colors to produce a continuous spectrum. This is much simpler than it sounds in this description; see the recipes in chapter 16 for an illustration of its use. If valueRange and colors are both NULL, a default grayscale color scheme will be used in SLiMgui, but an error will result if mapColor() is called.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other SpatialMap: SM, add(), blend(), changeValues(), divide(), exp(), gridValues(), interpolate(), mapColor(), mapImage(), mapValue(), multiply(), power(), range(), rescale(), sampleImprovedNearbyPoint(), sampleNearbyPoint(), smooth(), subtract()

---

changeValues *SLiM method changeValues*

Description

Documentation for SLiM function `changeValues`, which is a method of the SLiM class `SpatialMap`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
changeValues(x)
```

Arguments

`x`	An object of type integer or float or SpatialMap object. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 713.

Changes the grid values used for the target spatial map. The parameter x should be either a SpatialMap object from which values are taken directly, or a vector, matrix, or array of numeric values as described in the documentation for defineSpatialMap(). Other characteristics of the spatial map, such as its color mapping (if defined), its spatial bounds, and its spatiality, will remain unchanged. The grid resolution of the spatial map is allowed to change with this method. This method is useful for changing the values of a spatial map over time, such as to implement changes to the landscape's characteristics due to seasonality, climate change, processes such as fire or urbanization, and so forth. As with the original map values provided to defineSpatialMap(), it is often useful to read map values from a PNG image file using the Eidos class Image.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other SpatialMap: `SM`, `add()`, `blend()`, `changeColors()`, `divide()`, `exp()`, `gridValues()`, `interpolate()`, `mapColor()`, `mapImage()`, `mapValue()`, `multiply()`, `power()`, `range()`, `rescale()`, `sampleImprovedNearbyPoint()`, `sampleNearbyPoint()`, `smooth()`, `subtract()`

`clearKeysAndValues` *SLiM method clearKeysAndValues*

Description

Documentation for SLiM function `clearKeysAndValues`, which is a method of the SLiM class `LogFile`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
clearKeysAndValues(void)
```

Arguments

`void`	An object of type . See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 702.

This Dictionary method has an override in LogFile to make it illegal to call, since LogFile manages its Dictionary entries.

Value

An object of type void or logical.

Copyright

Author(s)

Benjamin C Haller (`<bhaller@benhaller.com>`) and Philipp W Messer (`<messer@cornell.edu>`)

See Also

Other LogFile: LF, addCustomColumn(), addCycleStage(), addCycle(), addKeysAndValuesFrom(), addMeanSDColumns(), addPopulationSexRatio(), addPopulationSize(), addSubpopulationSexRatio(), addSubpopulationSize(), addSuppliedColumn(), addTick(), flush(), logRow(), setFilePath(), setLogInterval(), setSuppliedValue(), setValue(), willAutolog()

---

clippedIntegral *SLiM method clippedIntegral*

Description

Documentation for SLiM function clippedIntegral, which is a method of the SLiM class InteractionType. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
clippedIntegral(receivers)
```

Arguments

receivers	An object of type null or Individual object. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 691.

Returns a vector containing the integral of the interaction function as experienced by each of the individuals in receivers. For each given individual, the interaction function is clipped to the edges of the spatial bounds of the subpopulation that individual inhabits; the individual's spatial position must be within bounds or an error is raised. A periodic boundary will, correctly, not clip the interaction function. The interaction function is also clipped to the interaction's maximum distance; that distance must be less than half of the extent of the spatial bounds in each dimension (so that, for a given dimension, the interaction function is clipped by the spatial bounds on only one side), otherwise an error is raised. Note that receiver constraints are not applied; an individual might not actually receive any interactions because of those constraints, but it is still considered to have the same interaction function integral. If receivers is NULL, the maximal integral is returned, as would be experienced by an individual farther than the maximum distance from any edge. The evaluate() method must have been previously called for the receiver subpopulation, and positions saved at evaluation time will be used. If the InteractionType is non-spatial, this method may not be called. The computed value of the integral is not exact; it is calculated by an approximate numerical method designed to be fast, but the error should be fairly small (typically less than 1 occur the first time this method is called (perhaps taking more than a second, depending upon hardware), but subsequent calls should be very fast. This method does not invoke interaction() callbacks; the calculated integrals are only for the

interaction function itself, and so will not be accurate if interaction() callbacks modify the relationship between distance and interaction strength. For this reason, the overhead of the first call will not reoccur when individuals move or when the interaction is re-evaluated; for typical models, the initial overhead will be incurred only once. The initial overhead will reoccur, however, if the interaction function itself, or the maximum interaction distance, are changed; frequent change of those parameters may render the performance of this method unacceptable. The integral values returned by clippedIntegral() can be useful for computing interaction metrics that are scaled by the amount of "interaction field" (to coin a term) that is present for a given individual, producing metrics of interaction density. Notably, the localPopulationDensity() method automatically incorporates the mechanics of clippedIntegral() into the calculations it performs; see that method's documentation for further discussion of this concept. This approach can also be useful with the interactingNeighborCount() method, provided that the interaction function is of type "f" (since the neighbor count does not depend upon interaction strength).

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other InteractionType: IT, distanceFromPoint(), distance(), drawByStrength(), evaluate(), interactingNeighborCount(), interactionDistance(), localPopulationDensity(), nearestInteractingNei nearestNeighborsOfPoint(), nearestNeighbors(), neighborCountOfPoint(), neighborCount(), setConstraints(), setInteractionFunction(), strength(), testConstraints(), totalOfNeighborStrengths unevaluate()

Co *Community*

Description

Documentation for Community class from SLiM

Details

This class represents the top level of a SLiM simulation: a community containing one or more species (represented by class Species). The community, represented by a global Community object with the variable name community, is responsible for execution of the tick cycle for active species in each tick, management of the script blocks associated with a simulation, and creation of log files (by creating and owning instances of the LogFile class). The community exists even in singlespecies models. This class has the following methods (functions):

- `createLogFile`
- `deregisterScriptBlock`
- `genomicElementTypesWithIDs`
- `interactionTypesWithIDs`
- `mutationTypesWithIDs`
- `outputUsage`
- `registerEarlyEvent`
- `registerFirstEvent`
- `registerInteractionCallback`
- `registerLateEvent`
- `rescheduleScriptBlock`
- `scriptBlocksWithIDs`
- `simulationFinished`
- `speciesWithIDs`
- `subpopulationsWithIDs`
- `usage`

This class has the following properties:

allGenomicElementTypes A property of type GenomicElementType object. This property is a constant, so it is not modifiable. **Property Description:** All of the GenomicElementType objects defined in the simulation.

allInteractionTypes A property of type InteractionType object. This property is a constant, so it is not modifiable. **Property Description:** All of the InteractionType objects defined in the simulation.

allMutationTypes A property of type MutationType object. This property is a constant, so it is not modifiable. **Property Description:** All of the MutationType objects defined in the simulation.

allScriptBlocks A property of type SLiMEidosBlock object. This property is a constant, so it is not modifiable. **Property Description:** All registered SLiMEidosBlock objects in the simulation.

allSpecies A property of type Species object. This property is a constant, so it is not modifiable. **Property Description:** All of the Species objects defined in the simulation (in species declaration order).

allSubpopulations A property of type Subpopulation object. This property is a constant, so it is not modifiable. **Property Description:** All of the Subpopulation objects defined in the simulation.

cycleStage A property of type string. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The current cycle stage, as a string. The values of this property essentially mirror the cycle stages of WF and nonWF models (see chapters 23 and 24). Common values include "first" (during execution of first() events), "early" (during execution of early() events), "reproduction" (during offspring generation), "fitness" (during fitness evaluation), "survival" (while applying selection and mortality in nonWF models), and "late" (during execution of late() events). Other possible values include "begin" (during internal setup before each cycle), "tally" (while tallying mutation reference counts and removing fixed mutations), "swap" (while swapping the offspring generation into the parental generation in WF models), "end" (during internal bookkeeping after each cycle), and "console" (during the in-between-ticks state in which commands in SLiMgui's Eidos console are executed). It would probably be a good idea not to use this latter set of values; they are probably not user-visible during ordinary model execution anyway. During execution of initialize() callbacks, no Community object yet exists and so this property cannot be accessed. To detect this state, use exists("community"); if that is F, community does not exist, and therefore your code is executing during initialize() callbacks (or outside of SLiM entirely, in some other Eidos-based context).

logFiles A property of type LogFile object. This property is a constant, so it is not modifiable. **Property Description:** The LogFile objects being used in the simulation.

modelType A property of type string. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The type of model being simulated, as specified in initializeSLiMModelType(). This will be "WF" for WF models (Wright-Fisher models, the default), or "nonWF" for nonWF models (non-Wright-Fisher models; see section 1.6 for discussion). This must be the same for all species in the community; it is therefore a property on Community, not Species.

tag A property of type integer. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** A user-defined integer value. The value of tag is initially undefined, and it is an error to try to read it; if you wish it to have a defined value, you must arrange that yourself by explicitly setting its value prior to using it elsewhere in your code. The value of tag is not used by SLiM; it is free for you to use. See also the getValue() and setValue() methods (provided by the Dictionary class; see the Eidos manual), for another way of attaching state to the simulation.

tick A property of type integer. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** The current tick number.

verbosity A property of type integer. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** The verbosity level, for SLiM's logging of information about the simulation. This is 1 by default, but can be changed at the command line with the -l[ong] option. It is provided here so that scripts can consult it to govern the level of verbosity of their own output, or set the verbosity level for particular sections of their code. A verbosity level of 0 suppresses most of SLiM's optional output; 2 adds some extra output beyond SLiM's standard output. See sections 20.3 and 21.4 for more information.

See Also

Other Community: createLogFile(), deregisterScriptBlock(), genomicElementTypesWithIDs(), interactionTypesWithIDs(), mutationTypesWithIDs(), outputUsage(), registerEarlyEvent(), registerFirstEvent(), registerInteractionCallback(), registerLateEvent(), rescheduleScriptBlock(), scriptBlocksWithIDs(), simulationFinished(), speciesWithIDs(), subpopulationsWithIDs(), usage()

---

code *Extract or set code from a* slimr_script *object*

Description

Extract or set code from a slimr_script object

Usage

```
code(x)

code(x) <- value
```

Arguments

x	A slimr_script object
value	Code to replace with.

Examples

```
script <- slim_script(
  slim_block_init_minimal(),
  slim_block_finish(100)
)
code(script)
```

---

codonsToAminoAcids *SLiM method codonsToAminoAcids*

Description

Documentation for SLiM function codonsToAminoAcids, which is a method of the SLiM class SLiMBuiltin. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
codonsToAminoAcids(codons, long, paste)
```

Arguments

`codons`	An object of type integer. See details for description.
`long`	An object of type logical or integer. Must be of length 1 (a singleton). The default value is `F`. See details for description.
`paste`	An object of type logical. Must be of length 1 (a singleton). The default value is `T`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 748.

Returns the amino acid sequence corresponding to the codon sequence in codons. Codons should be represented with values in [0, 63] where AAA is 0, AAC is 1, AAG is 2, and TTT is 63; see ancestralNucleotides() for discussion of this encoding. If long is F (the default), the standard single-letter codes for amino acids will be used (where Serine is "S", etc.); if long is T, the standard three-letter codes will be used instead (where Serine is "Ser", etc.). Beginning in SLiM 3.5, if long is 0, integer codes will be used as follows (and paste will be ignored): stop (TAA, TAG, TGA) 0 Alanine 1 Arginine 2 Asparagine 3 Aspartic acid (Aspartate) 4 Cysteine 5 Glutamine 6 Glutamic acid (Glutamate) 7 Glycine 8 Histidine 9 Isoleucine 10 Leucine 11 Lysine 12 Methionine 13 Phenylalanine 14 Proline 15 Serine 16 Threonine 17 Tryptophan 18 Tyrosine 19 Valine 20 There does not seem to be a widely used standard for integer coding of amino acids, so SLiM just numbers them alphabetically, making stop codons 0. If you want a different coding, you can make your own 64-element vector and use it to convert codons to whatever integer codes you need. Other integer values of long are reserved for future use (to support other codings), and will currently produce an error. When long is T or F and paste is T (the default), the amino acid sequence returned will be a singleton string, such as "LYATI" (when long is F) or "Leu-Tyr-Ala-Thr-Ile" (when long is T). When long is T or F and paste is F, the amino acid sequence will instead be returned as a string vector, with one element per amino acid, such as "L" "Y" "A" "T" "I" (when long is F) or "Leu" "Tyr" "Ala" "Thr" "Ile" (when long is T). Using the paste=T option is considerably faster than using paste() in script. This function interprets the supplied codon sequence as the sense strand (i.e., the strand that is not transcribed, and which mirrors the mRNA's sequence). This uses the standard DNA codon table directly. For example, if the nucleotide sequence is CAA TTC, that will correspond to a codon vector of 16 61, and will result in the amino acid sequence Gln-Phe ("QF"). (is)codonsToNucleotides(integer codons, [string$ format = "string"]) Returns the nucleotide sequence corresponding to the codon sequence supplied in codons. Codons should be represented with values in [0, 63] where AAA is 0, AAC is 1, AAG is 2, and TTT is 63; see ancestralNucleotides() for discussion of this encoding. The format parameter controls the format of the returned sequence. It may be "string" to obtain the sequence as a singleton string (e.g., "TATACG"), "char" to obtain it as a string vector of single characters (e.g., "T", "A", "T", "A", "C", "G"), or "integer" to obtain it as an integer vector (e.g., 3, 0, 3, 0, 1, 2), using SLiM's standard code of A=0, C=1, G=2, T=3.

Value

An object of type integer or string.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other SLiMBuiltin: SB, calcFST(), calcHeterozygosity(), calcInbreedingLoad(), calcPairHeterozygosity(
calcVA(), calcWattersonsTheta(), mm16To256(), mmJukesCantor(), mmKimura(), nucleotideCounts(),
nucleotideFrequencies(), nucleotidesToCodons(), summarizeIndividuals(), treeSeqMetadata()

Examples

```
## This just brings up the documentation:
codonsToAminoAcids()
```

---

configureDisplay *SLiM method configureDisplay*

Description

Documentation for SLiM function `configureDisplay`, which is a method of the SLiM class Subpopulation. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
configureDisplay(center, scale, color)
```

Arguments

center	An object of type null or float. The default value is NULL. See details for description.
scale	An object of type null or float. Must be of length 1 (a singleton). The default value is NULL. See details for description.
color	An object of type null or string. Must be of length 1 (a singleton). The default value is NULL. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 738.

This method customizes the display of the subpopulation in SLiMgui's Population Visualization graph. When this method is called by a model running outside SLiMgui, it will do nothing except typechecking and bounds-checking its arguments. When called by a model running in SLiMgui, the position, size, and color of the subpopulation's displayed circle can be controlled as specified below. The center parameter sets the coordinates of the center of the subpopulation's displayed circle; it must be a float vector of length two, such that center[0] provides the x-coordinate and center[1] provides the y-coordinate. The square central area of the Population Visualization occupies scaled coordinates in [0,1] for both x and y, so the values in center must be within those bounds. If a value of NULL is provided, SLiMgui's default center will be used (which currently arranges subpopulations in a circle). The scale parameter sets a scaling factor to be applied to the radius of the subpopulation's displayed circle. The default radius used by SLiMgui is a function of the subpopulation's number of individuals; this default radius is then multiplied by scale. If a value of NULL is provided, the default radius will be used; this is equivalent to supplying a scale of 1.0. Typically the same scale value should be used by all subpopulations, to scale all of their circles up or down uniformly, but that is not required. The color parameter sets the color to be used for the displayed subpopulation's circle. Colors may be specified by name, or with hexadecimal RGB values of the form "#RRGGBB" (see the Eidos manual). If color is NULL or the empty string, "", SLiMgui's default (fitness-based) color will be used.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Subpopulation: P, addCloned(), addCrossed(), addEmpty(), addRecombinant(), addSelfed(), addSpatialMap(), cachedFitness(), defineSpatialMap(), outputMSSample(), outputSample(), outputVCFSample(), pointDeviated(), pointInBounds(), pointPeriodic(), pointReflected(), pointStopped(), pointUniform(), removeSpatialMap(), removeSubpopulation(), sampleIndividuals(), setCloningRate(), setMigrationRates(), setSelfingRate(), setSexRatio(), setSpatialBounds(), setSubpopulationSize(), spatialMapColor(), spatialMapImage(), spatialMapValue(), subsetIndividuals(), takeMigrants()

`containsMarkerMutation`

SLiM method containsMarkerMutation

Description

Documentation for SLiM function `containsMarkerMutation`, which is a method of the SLiM class `Genome`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
containsMarkerMutation(mutType, position, returnMutation)
```

Arguments

`mutType`	An object of type integer or MutationType object. Must be of length 1 (a singleton). See details for description.
`position`	An object of type integer. Must be of length 1 (a singleton). See details for description.
`returnMutation`	An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 672.

Returns T if the genome contains a mutation of type mutType at position, F otherwise (if returnMutation has its default value of F; see below). This method is, as its name suggests, intended for checking for "marker mutations": mutations of a special mutation type that are not literally mutations in the usual sense, but instead are added in to particular genomes to mark them as possessing some property. Marker mutations are not typically added by SLiM's mutation-generating machinery; instead they are added explicitly with addNewMutation() or addNewDrawnMutation() at a known, constant position in the genome. This method provides a check for whether a marker mutation of a given type exists in a particular genome; because the position to check is known in advance, that check can be done much faster than the equivalent check with containsMutations() or countOfMutationsOfType(), using a binary search of the genome. See section 14.4 for one example of a model that uses marker mutations - in that case, to mark chromosomes that possess an inversion. If returnMutation is T (an option added in SLiM 3), this method returns the actual mutation found, rather than just T or F. More specifically, the first mutation found of mutType at position will be returned; if more than one such mutation exists in the target genome, which one is returned is not defined. If returnMutation is T and no mutation of mutType is found at position, NULL will be returned.

Value

An object of type null or logical or Mutation object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Genome: G, addMutations(), addNewDrawnMutation(), addNewMutation(), containsMutations(), countOfMutationsOfType(), mutationCountsInGenomes(), mutationFrequenciesInGenomes(), mutationsOfType(), nucleotides(), outputMS(), outputVCF(), output(), positionsOfMutationsOfType(), readFromMS(), readFromVCF(), removeMutations(), sumOfMutationsOfType()

---

containsMutations *SLiM method containsMutations*

Description

Documentation for SLiM function `containsMutations`, which is a method of the SLiM class Genome. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Documentation for SLiM function `containsMutations`, which is a method of the SLiM class Individual. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
containsMutations(mutations)

containsMutations(mutations)
```

Arguments

mutations	An object of type Mutation object. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 672.

Returns a logical vector indicating whether each of the mutations in mutations is present in the genome; each element in the returned vector indicates whether the corresponding mutation is present (T) or absent (F). This method is provided for speed; it is much faster than the corresponding Eidos code.

Documentation for this function can be found in the official SLiM manual: page 682.

Returns a logical vector indicating whether each of the mutations in mutations is present in the individual (in either of its genomes); each element in the returned vector indicates whether the corresponding mutation is present (T) or absent (F). This method is provided for speed; it is much faster than the corresponding Eidos code.

Value

An object of type logical.

An object of type logical.

Copyright

This is documentation for a function in the SLiM software, and has been reproduced from the official manual, which can be found here: http://benhaller.com/slim/SLiM_Manual.pdf. This documentation is Copyright © 2016-2020 Philipp Messer. All rights reserved. More information about SLiM can be found on the official website: https://messerlab.org/slim/

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Genome: G, addMutations(), addNewDrawnMutation(), addNewMutation(), containsMarkerMutation(), countOfMutationsOfType(), mutationCountsInGenomes(), mutationFrequenciesInGenomes(), mutationsOfType(), nucleotides(), outputMS(), outputVCF(), output(), positionsOfMutationsOfType(), readFromMS(), readFromVCF(), removeMutations(), sumOfMutationsOfType()

Other Individual: In, countOfMutationsOfType(), relatedness(), setSpatialPosition(), sharedParentCount(), sumOfMutationsOfType(), uniqueMutationsOfType()

`countOfMutationsOfType`
SLiM method countOfMutationsOfType

Description

Documentation for SLiM function `countOfMutationsOfType`, which is a method of the SLiM class `Genome`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Documentation for SLiM function `countOfMutationsOfType`, which is a method of the SLiM class `Individual`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Documentation for SLiM function `countOfMutationsOfType`, which is a method of the SLiM class `Species`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
countOfMutationsOfType(mutType)

countOfMutationsOfType(mutType)

countOfMutationsOfType(mutType)
```

Arguments

`mutType`	An object of type integer or MutationType object. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 672.

Returns the number of mutations that are of the type specified by mutType, out of all of the mutations in the genome. If you need a vector of the matching Mutation objects, rather than just a count, use -mutationsOfType(). This method is provided for speed; it is much faster than the corresponding Eidos code.

Documentation for this function can be found in the official SLiM manual: page 683.

Returns the number of mutations that are of the type specified by mutType, out of all of the mutations in the individual (in both of its genomes; a mutation that is present in both genomes counts twice). If you need a vector of the matching Mutation objects, rather than

just a count, you should probably use uniqueMutationsOfType(). This method is provided for speed; it is much faster than the corresponding Eidos code.

Documentation for this function can be found in the official SLiM manual: page 720.

Returns the number of mutations that are of the type specified by mutType, out of all of the mutations that are currently active in the species. If you need a vector of the matching Mutation objects, rather than just a count, use -mutationsOfType(). This method is often used to determine whether an introduced mutation is still active (as opposed to being either lost or fixed). This method is provided for speed; it is much faster than the corresponding Eidos code.

Value

An object of type integer. Return will be of length 1 (a singleton)

An object of type integer. Return will be of length 1 (a singleton)

An object of type integer. Return will be of length 1 (a singleton)

Copyright

This is documentation for a function in the SLiM software, and has been reproduced from the official manual, which can be found here: http://benhaller.com/slim/SLiM_Manual.pdf. This documentation is Copyright © 2016-2020 Philipp Messer. All rights reserved. More information about SLiM can be found on the official website: https://messerlab.org/slim/

This is documentation for a function in the SLiM software, and has been reproduced from the official manual, which can be found here: http://benhaller.com/slim/SLiM_Manual.pdf. This documentation is Copyright © 2016-2020 Philipp Messer. All rights reserved. More information about SLiM can be found on the official website: https://messerlab.org/slim/

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Genome: G, addMutations(), addNewDrawnMutation(), addNewMutation(), containsMarkerMutation(), containsMutations(), mutationCountsInGenomes(), mutationFrequenciesInGenomes(), mutationsOfType(), nucleotides(), outputMS(), outputVCF(), output(), positionsOfMutationsOfType(), readFromMS(), readFromVCF(), removeMutations(), sumOfMutationsOfType()

Other Individual: In, containsMutations(), relatedness(), setSpatialPosition(), sharedParentCount(), sumOfMutationsOfType(), uniqueMutationsOfType()

Other Species: Sp, addSubpopSplit(), addSubpop(), individualsWithPedigreeIDs(), killIndividuals(), mutationCounts(), mutationFrequencies(), mutationsOfType(), outputFixedMutations(), outputFull(), outputMutations(), readFromPopulationFile(), recalculateFitness(), registerFitnessEffectCallback(), registerMateChoiceCallback(), registerModifyChildCallback(), registerMutationCallback(), registerMutationEffectCallback(), registerRecombinationCallback(), registerReproductionCallback(), registerSurvivalCallback(), simulationFinished(), skipTick(), subsetMutations(), treeSeqCoalesced(), treeSeqOutput(), treeSeqRememberIndividuals(), treeSeqSimplify()

`createLogFile` *SLiM method createLogFile*

Description

Documentation for SLiM function `createLogFile`, which is a method of the SLiM class `Community`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
createLogFile(
  filePath,
  initialContents,
  append,
  compress,
  sep,
  logInterval,
  flushInterval
)
```

Arguments

`filePath`	An object of type string. Must be of length 1 (a singleton). See details for description.
`initialContents`	An object of type null or string. The default value is `NULL`. See details for description.
`append`	An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.
`compress`	An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.
`sep`	An object of type string. Must be of length 1 (a singleton). The default value is `","`. See details for description.

`logInterval` An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

`flushInterval` An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 665.

Creates and returns a new LogFile object that logs data from the simulation (see the documentation for the LogFile class for details). Logged data will be written to the file at filePath, overwriting any existing file at that path by default, or appending to it instead if append is T (successive rows of the log table will always be appended to the previously written content, of course). Before the header line for the log is written out, any string elements in initialContents will be written first, separated by newlines, allowing for a user-defined file header. If compress is T, the contents will be compressed with zlib as they are written, and the standard .gz extension for gzip-compressed files will be appended to the filename in filePath if it is not already present. The sep parameter specifies the separator between data values within a row. The default of "," will generate a "comma-separated value" (CSV) file, while passing sep="\t" will use a tab separator instead to generate a "tab-separated value" (TSV) file. Other values for sep may also be used, but are less standard. LogFile supports periodic automatic logging of a new row of data, enabled by supplying a non-NULL value for logInterval. In this case, a new row will be logged (as if logRow() were called on the LogFile) at the end of every logInterval ticks (just before the tick counter increments, in both WF and nonWF models), starting at the end of the tick in which the LogFile was created. A logInterval of 1 will cause automatic logging at the end of every tick, whereas a logInterval of NULL disables automatic logging. Automatic logging can always be disabled or reconfigured later with the LogFile method setLogInterval(), or logging can be triggered manually by calling logRow(). When compression is enabled, LogFile flushes new data lazily by default, for performance reasons, buffering data for multiple rows before writing to disk. Passing a non-NULL value for flushInterval requests a flush every flushInterval rows (with a value of 1 providing unbuffered operation). Note that flushing very frequently will likely result in both lower performance and a larger final file size (in one simple test, 48943 bytes instead of 4280 bytes, or more than a 10× increase in size). Alternatively, passing a very large value for flushInterval will effectively disable automatic flushing, except at the end of the simulation (but be aware that this may use a large amount of memory for large log files). In any case, the log file will be created immediately, with its requested initial contents; the initial write is not buffered. When compression is not enabled, the flushInterval setting is ignored. The LogFile documentation discusses how to configure and use LogFile to write out the data you are interested in from your simulation; see section 25.10.

Value

An object of type LogFile object. Return will be of length 1 (a singleton)

Copyright

pdf. This documentation is Copyright © 2016-2020 Philipp Messer. All rights reserved. More information about SLiM can be found on the official website: https://messerlab.org/slim/

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Community: Co, deregisterScriptBlock(), genomicElementTypesWithIDs(), interactionTypesWithID mutationTypesWithIDs(), outputUsage(), registerEarlyEvent(), registerFirstEvent(), registerInteractionCallback(), registerLateEvent(), rescheduleScriptBlock(), scriptBlocksWithIDs() simulationFinished(), speciesWithIDs(), subpopulationsWithIDs(), usage()

---

defineSpatialMap *SLiM method defineSpatialMap*

Description

Documentation for SLiM function `defineSpatialMap`, which is a method of the SLiM class Subpopulation. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
defineSpatialMap(name, spatiality, values, interpolate, valueRange, colors)
```

Arguments

`name`	An object of type string. Must be of length 1 (a singleton). See details for description.
`spatiality`	An object of type string. Must be of length 1 (a singleton). See details for description.
`values`	An object of type numeric. See details for description.
`interpolate`	An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.
`valueRange`	An object of type null or integer or float. The default value is `NULL`. See details for description.
`colors`	An object of type null or string. The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 738.

Defines a spatial map for the subpopulation; see the SpatialMap documentation regarding this class. The new map is automatically added to the subpopulation; addSpatialMap() does not need to be called. (That method is for sharing the map with additional subpopulations, beyond the one for which the map was originally defined.) The new SpatialMap object is returned, and may be retained permanently using defineConstant() or defineGlobal() for convenience. The name of the map is given by name, and can be used to identify it. The map uses the spatial dimensions referenced by spatiality, which must be a subset of the dimensions defined for the simulation in initializeSLiMOptions(). Spatiality "x" is permitted for dimensionality "x"; spatiality "x", "y", or "xy" for dimensionality "xy"; and spatiality "x", "y", "z", "xy", "yz", "xz", or "xyz" for dimensionality "xyz". The spatial map is defined by a grid of values supplied in parameter values. That grid of values is aligned with the spatial bounds of the subpopulation, as described in more detail below; the spatial map is therefore coupled to those spatial bounds, and can only be used in subpopulations that match those particular spatial bounds (to avoid stretching or shrinking the map). The remaining optional parameters are described below. Note that the semantics of this method changed in SLiM 3.5; in particular, the gridSize parameter was removed, and the interpretation of the values parameter changed as described below. Existing code written prior to SLiM 3.5 will produce an error, due to the removed gridSize parameter, and must be revised carefully to obtain the same result, even if NULL had been passed for gridSize previously. Beginning in SLiM 3.5, the values parameter must be a vector/matrix/array with the number of dimensions appropriate for the declared spatiality of the map; for example, a map with spatiality "x" would require a (one-dimensional) vector, spatiality "xy" would require a (two-dimensional) matrix, and a map with spatiality of "xyz" would require a three-dimensional array. (See the Eidos manual for discussion of vectors, matrices, and arrays.) The data in values is interpreted in such a way that a two-dimensional matrix of values, with (0, 0) at upper left and values by column, is transformed into the format expected by SLiM, with (0, 0) at lower left and values by row; in other words, the twodimensional matrix as it prints in the Eidos console will match the appearance of the two-dimensional spatial map as seen in SLiMgui. This is a change in behavior from versions prior to SLiM 3.5; it ensures that images loaded from disk with the Eidos class Image can be used directly as spatial maps, achieving the expected orientation, with no need for transposition or flipping. If the spatial map is a three-dimensional array, it is read as successive z-axis "planes", each of which is a two-dimensional matrix that is treated as described above. Moving on to the other parameters of defineSpatialMap(): if interpolate is F, values across the spatial map are not interpolated; the value at a given point is equal to the nearest value defined by the grid of values specified. If interpolate is T, values across the spatial map will be interpolated (using linear, bilinear, or trilinear interpolation as appropriate) to produce spatially continuous variation in values. In either case, the corners of the value grid are exactly aligned with the corners of the spatial boundaries of the subpopulation as specified by setSpatialBounds(), and the value grid is then stretched across the spatial extent of the subpopulation in such a manner as to produce equal spacing between the values along each dimension. The setting of interpolation only affects how values between these grid points are calculated: by nearest-neighbor, or by linear interpolation. Interpolation of spatial maps with periodic boundaries is not handled specially; to ensure that the edges of a periodic spatial map join smoothly, simply ensure that the grid values at the edges of the map are identical, since they will be coincident after periodic wrapping. Note that

cubic/bicubic interpolation is generally smoother than linear/bilinear interpolation, with fewer artifacts, but it is substantially slower to calculate; use the interpolate() method of SpatialMap to precalculate an interpolated map using cubic/bucubic interpolation. The valueRange and colors parameters travel together; either both are unspecified, or both are specified. They control how map values will be transformed into colors, by SLiMgui and by the mapColor() method. The valueRange parameter establishes the color-mapped range of spatial map values, as a vector of length two specifying a minimum and maximum; this does not need to match the actual range of values in the map. The colors parameter then establishes the corresponding colors for values within the interval defined by valueRange: values less than or equal to valueRange[0] will map to colors[0], values greater than or equal to valueRange[1] will map to the last colors value, and intermediate values will shade continuously through the specified vector of colors, with interpolation between adjacent colors to produce a continuous spectrum. This is much simpler than it sounds in this description; see the recipes in chapter 16 for an illustration of its use. Note that at present, SLiMgui will only display spatial maps of spatiality "x", "y", or "xy"; the colormapping parameters will simply be ignored by SLiMgui for other spatiality values (even if the spatiality is a superset of these values; SLiMgui will not attempt to display an "xyz" spatial map, for example, since it has no way to choose which 2D slice through the xyz space it ought to display). The mapColor() method will return translated color strings for any spatial map, however, even if SLiMgui is unable to display the spatial map. If there are multiple spatial maps that SLiMgui is capable of displaying, it choose one for display by default, but other maps may be selected from the action menu on the individuals view (by clicking on the button with the gear icon).

Value

An object of type SpatialMap object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Subpopulation: P, addCloned(), addCrossed(), addEmpty(), addRecombinant(), addSelfed(), addSpatialMap(), cachedFitness(), configureDisplay(), outputMSSample(), outputSample(), outputVCFSample(), pointDeviated(), pointInBounds(), pointPeriodic(), pointReflected(), pointStopped(), pointUniform(), removeSpatialMap(), removeSubpopulation(), sampleIndividuals(), setCloningRate(), setMigrationRates(), setSelfingRate(), setSexRatio(), setSpatialBounds(), setSubpopulationSize(), spatialMapColor(), spatialMapImage(), spatialMapValue(), subsetIndividuals(), takeMigrants()

`deregisterScriptBlock`

SLiM method deregisterScriptBlock

Description

Documentation for SLiM function `deregisterScriptBlock`, which is a method of the SLiM class `Community`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
deregisterScriptBlock(scriptBlocks)
```

Arguments

`scriptBlocks`	An object of type integer or SLiMEidosBlock object. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 666.

All SLiMEidosBlock objects specified by scriptBlocks (either with SLiMEidosBlock objects or with integer identifiers) will be scheduled for deregistration. The deregistered blocks remain valid, and may even still be executed in the current stage of the current tick (see section 26.11); the blocks are not actually deregistered and deallocated until sometime after the currently executing script block has completed. To immediately prevent a script block from executing, even when it is scheduled to execute in the current stage of the current tick, use the active property of the script block (see sections 25.12.1 and 26.11).

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Community: `Co`, `createLogFile()`, `genomicElementTypesWithIDs()`, `interactionTypesWithIDs()`, `mutationTypesWithIDs()`, `outputUsage()`, `registerEarlyEvent()`, `registerFirstEvent()`, `registerInteractionCallback()`, `registerLateEvent()`, `rescheduleScriptBlock()`, `scriptBlocksWithIDs()`, `simulationFinished()`, `speciesWithIDs()`, `subpopulationsWithIDs()`, `usage()`

`distance` *SLiM method distance*

Description

Documentation for SLiM function `distance`, which is a method of the SLiM class `InteractionType`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
distance(receiver, exerters)
```

Arguments

`receiver`	An object of type Individual object. Must be of length 1 (a singleton). See details for description.
`exerters`	An object of type null or Individual object. The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 692.

Returns a vector containing distances between receiver and the individuals in exerters. If exerters is NULL (the default), then a vector of the distances from receiver to all individuals in its subpopulation (including itself) is returned; this case may be handled differently internally, for greater speed, so supplying NULL is preferable to supplying the vector of all individuals in the subpopulation explicitly. Otherwise, all individuals in exerters must belong to a single subpopulation (but not necessarily the same subpopulation as receiver). The evaluate() method must have been previously called for the receiver and exerter subpopulations, and positions saved at evaluation time will be used. If the InteractionType is non-spatial, this method may not be called. Importantly, distances are calculated according to the spatiality of the InteractionType (as declared in initializeInteractionType()), not the dimensionality of the model as a whole (as declared in initializeSLiMOptions()). The distances returned are therefore the distances that would be used to calculate interaction strengths. However, distance() will return finite distances for all pairs of individuals, even if the individuals are non-interacting due to the maximum interaction distance or the interaction constraints; the distance() between an individual and itself will thus be 0. See interactionDistance() for an alternative distance definition.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other InteractionType: IT, clippedIntegral(), distanceFromPoint(), drawByStrength(), evaluate(), interactingNeighborCount(), interactionDistance(), localPopulationDensity(), nearestInteractingNeighbors(), nearestNeighborsOfPoint(), nearestNeighbors(), neighborCountOfPoint(), neighborCount(), setConstraints(), setInteractionFunction(), strength(), testConstraints(), totalOfNeighborStrengths(), unevaluate()

---

distanceFromPoint *SLiM method distanceFromPoint*

Description

Documentation for SLiM function distanceFromPoint, which is a method of the SLiM class InteractionType. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
distanceFromPoint(point, exerters)
```

Arguments

point	An object of type float. See details for description.
exerters	An object of type Individual object. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 692.

Returns a vector containing distances between the point given by the spatial coordinates in point, which may be thought of as the "receiver", and individuals in exerters. The point vector is interpreted as providing coordinates precisely as specified by the spatiality of the interaction type; if the interaction type's spatiality is "xz", for example, then point[0] is assumed to be an x value, and point[1] is assumed to be a z value. Be careful; this means that in general it is not safe to pass an individual's spatialPosition property for point, for example (although it is safe if the spatiality of the interaction matches the dimensionality of the simulation). A coordinate for a periodic spatial dimension must be within the spatial bounds for that dimension, since coordinates outside of periodic bounds are meaningless (pointPeriodic() may be used to ensure this); coordinates for non-periodic spatial dimensions are not restricted. All individuals in exerters must belong to a single subpopulation; the evaluate() method must have been previously called for that subpopulation, and positions saved at evaluation time will be used. If the InteractionType is non-spatial, this method may not be called. The vector point must be exactly as long as the spatiality of the InteractionType. Importantly, distances are calculated according to the spatiality of the InteractionType (as declared in initializeInteractionType()) not the dimensionality of the model as a whole (as declared in initializeSLiMOptions()). The distances are therefore interaction distances: the distances that are used to calculate interaction strengths. However, the maximum interaction distance and interaction constraints are not used. This method replaces the distanceToPoint() method that existed prior to SLiM 4.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other InteractionType: IT, clippedIntegral(), distance(), drawByStrength(), evaluate(), interactingNeighborCount(), interactionDistance(), localPopulationDensity(), nearestInteractingNei nearestNeighborsOfPoint(), nearestNeighbors(), neighborCountOfPoint(), neighborCount(), setConstraints(), setInteractionFunction(), strength(), testConstraints(), totalOfNeighborStrengths unevaluate()

`divide` *SLiM method divide*

Description

Documentation for SLiM function `divide`, which is a method of the SLiM class `SpatialMap`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
divide(x)
```

Arguments

`x`	An object of type integer or float or SpatialMap object. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 714.

Divides the spatial map by x. One possibility is that x is a singleton integer or float value; in this case, each grid value of the target spatial map is divided by x. Another possibility is that x is an integer or float vector/matrix/array of the same dimensions as the target spatial map's grid; in this case, each grid value of the target spatial map is divided by the corresponding value of x. The third possibility is that x is itself a (singleton) spatial map; in this case, each grid value of the target spatial map is divided by the corresponding grid value of x (and thus the two spatial maps must match in their spatiality, their spatial bounds, and their grid dimensions). The target spatial map is returned, to allow easy chaining of operations.

Value

An object of type SpatialMap object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other SpatialMap: `SM`, `add()`, `blend()`, `changeColors()`, `changeValues()`, `exp()`, `gridValues()`, `interpolate()`, `mapColor()`, `mapImage()`, `mapValue()`, `multiply()`, `power()`, `range()`, `rescale()`, `sampleImprovedNearbyPoint()`, `sampleNearbyPoint()`, `smooth()`, `subtract()`

`drawBreakpoints` *SLiM method drawBreakpoints*

Description

Documentation for SLiM function `drawBreakpoints`, which is a method of the SLiM class `Chromosome`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
drawBreakpoints(parent, n)
```

Arguments

`parent`	An object of type null or Individual object. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`n`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 662.

Draw recombination breakpoints, using the chromosome's recombination rate map, the current gene conversion parameters, and (in some cases - see below) any active and applicable recombination() callbacks. The number of breakpoints to generate, n, may be supplied; if it is NULL (the default), the number of breakpoints will be drawn based upon the overall recombination rate and the chromosome length (following the standard procedure in SLiM). Note that if the double-stranded breaks model has been chosen, the number of breakpoints generated will probably not be equal to the number requested, because most breakpoints will entail gene conversion tracts, which entail additional crossover breakpoints. It is generally recommended that the parent individual be supplied to this method, but parent is NULL by default. The individual supplied in parent is used for two purposes. First, in sexual models that define separate recombination rate maps for males versus females, the sex of parent will be used to determine which map is used; in this case, a non-NULL value must be supplied for parent, since the choice of recombination rate map must be determined. Second, in models that define recombination() callbacks, parent is used to determine the various pseudo-parameters that are passed to recombination() callbacks (individual, genome1, genome2, subpop), and the subpopulation to which parent belongs is

used to select which recombination() callbacks are applicable; given the necessity of this information, recombination() callbacks will not be called as a side effect of this method if parent is NULL. Apart from these two uses, parent is not used, and the caller does not guarantee that the generated breakpoints will actually be used to recombine the genomes of parent in particular. If a recombination() callback is called, genome1 for that callback will always be parent.genome1; in other words, drawBreakpoints() will always treat parent.genome1 as the initial copy strand. If the caller wishes to randomly choose an initial copy strand (which is usually desirable), they should do that themselves.

Value

An object of type integer.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Chromosome: Ch, ancestralNucleotides(), setAncestralNucleotides(), setGeneConversion(), setHotspotMap(), setMutationRate(), setRecombinationRate()

---

drawByStrength *SLiM method drawByStrength*

Description

Documentation for SLiM function drawByStrength, which is a method of the SLiM class InteractionType. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
drawByStrength(receiver, count, exerterSubpop, returnDict)
```

Arguments

`receiver`	An object of type Individual object. See details for description.
`count`	An object of type integer. Must be of length 1 (a singleton). The default value is `1`. See details for description.
`exerterSubpop`	An object of type null or Subpopulation object. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`returnDict`	An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 692.

Returns an object<Individual> vector containing up to count individuals drawn from exerterSubpop, or if that is NULL (the default), then from the subpopulation of receiver, which must be singleton in the default mode of operation (but see below). The probability of drawing particular individuals is proportional to the strength of interaction they exert upon receiver (which is zero for receiver itself). All exerters must belong to a single subpopulation (but not necessarily the same subpopulation as receiver). The evaluate() method must have been previously called for the receiver and exerter subpopulations, and positions saved at evaluation time will be used. This method may be used with either spatial or non-spatial interactions, but will be more efficient with spatial interactions that set a short maximum interaction distance. Draws are done with replacement, so the same individual may be drawn more than once; sometimes using unique() on the result of this call is therefore desirable. If more than one draw will be needed, it is much more efficient to use a single call to drawByStrength(), rather than drawing individuals one at a time. Note that if no individuals exert a non-zero interaction strength upon receiver, the vector returned will be zerolength; it is important to consider this possibility. Beginning in SLiM 4.1, this method has a vectorized mode of operation in which the receiver parameter may be non-singleton. To switch the method to this mode, pass T for returnDict, rather than the default of F (the operation of which is described above). In this mode, the return value is a Dictionary object instead of a vector of Individual objects. This dictionary uses integer keys that range from 0 to N-1, where N is the number of individuals passed in receiver; these keys thus correspond directly to the indices of the individuals in receiver, and there is one entry in the dictionary for each receiver. The value in the dictionary, for a given integer key, is an object<Individual> vector with the individuals drawn for the corresponding receiver, exactly as described above for the non-vectorized case. The results for each receiver can therefore be obtained from the returned dictionary with getValue(), passing the index of the receiver. The speed of this mode of operation will probably be similar to the speed of making N separate non-vectorized calls to drawByStrength(), but may have other advantages. In this mode of operation, all receivers must belong to the same subpopulation.

Value

An object of type .

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other InteractionType: IT, clippedIntegral(), distanceFromPoint(), distance(), evaluate(), interactingNeighborCount(), interactionDistance(), localPopulationDensity(), nearestInteractingNei nearestNeighborsOfPoint(), nearestNeighbors(), neighborCountOfPoint(), neighborCount(), setConstraints(), setInteractionFunction(), strength(), testConstraints(), totalOfNeighborStrengths unevaluate()

---

drawSelectionCoefficient

SLiM method drawSelectionCoefficient

Description

Documentation for SLiM function `drawSelectionCoefficient`, which is a method of the SLiM class `MutationType`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
drawSelectionCoefficient(n)
```

Arguments

`n`	An object of type integer. Must be of length 1 (a singleton). The default value is `1`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 709.

Draws and returns a vector of n selection coefficients using the currently defined distribution of fitness effects (DFE) for the target mutation type. See section 25.11 above for discussion of the supported distributions and their uses. If the DFE is type "s", this method will result in synchronous execution of the DFE's script.

Value

An object of type float or void.

Copyright

This is documentation for a function in the SLiM software, and has been reproduced from the official manual, which can be found here: http://benhaller.com/slim/SLiM_Manual.pdf. This documentation is Copyright © 2016-2020 Philipp Messer. All rights reserved. More information about SLiM can be found on the official website: https://messerlab.org/slim/

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other MutationType: MT, setDistribution()

E *Eidos*

Description

Documentation for Eidos class from SLiM

Details

NA This class has the following methods (functions):

- eidos_abs
- eidos_acos
- eidos_asin
- eidos_atan
- eidos_atan2
- eidos_ceil
- eidos_cos
- eidos_cumProduct
- eidos_cumSum
- eidos_exp
- eidos_floor
- eidos_integerDiv
- eidos_integerMod
- eidos_isFinite

- eidos_isInfinite
- eidos_isNAN
- eidos_log
- eidos_log10
- eidos_log2
- eidos_product
- eidos_round
- eidos_setDifference
- eidos_setIntersection
- eidos_setSymmetricDifference
- eidos_setUnion
- eidos_sin
- eidos_sqrt
- eidos_sum
- eidos_sumExact
- eidos_tan
- eidos_trunc
- eidos_cor
- eidos_cov
- eidos_max
- eidos_mean
- eidos_min
- eidos_pmax
- eidos_pmin
- eidos_quantile
- eidos_range
- eidos_rank
- eidos_sd
- eidos_ttest
- eidos_var
- eidos_dmvnorm
- eidos_dbeta
- eidos_dexp
- eidos_dgamma
- eidos_dnorm
- eidos_findInterval
- eidos_pnorm

- eidos_qnorm
- eidos_rbeta
- eidos_rbinom
- eidos_rcauchy
- eidos_rdunif
- eidos_rexp
- eidos_rf
- eidos_rgamma
- eidos_rgeom
- eidos_rlnorm
- eidos_rmvnorm
- eidos_rnbinom
- eidos_rnorm
- eidos_rpois
- eidos_runif
- eidos_rweibull
- eidos_c
- eidos_float
- eidos_integer
- eidos_logical
- eidos_object
- eidos_rep
- eidos_repEach
- eidos_sample
- eidos_seq
- eidos_seqAlong
- eidos_seqLen
- eidos_string
- eidos_all
- eidos_any
- eidos_cat
- eidos_catn
- eidos_format
- eidos_identical
- eidos_ifelse
- eidos_length
- eidos_match

- eidos_order
- eidos_paste
- eidos_paste0
- eidos_print
- eidos_rev
- eidos_size
- eidos_sort
- eidos_sortBy
- eidos_str
- eidos_tabulate
- eidos_unique
- eidos_which
- eidos_whichMax
- eidos_whichMin
- eidos_asFloat
- eidos_asInteger
- eidos_asLogical
- eidos_asString
- eidos_elementType
- eidos_isFloat
- eidos_isInteger
- eidos_isLogical
- eidos_isNULL
- eidos_isObject
- eidos_isString
- eidos_type
- eidos_nchar
- eidos_strcontains
- eidos_strfind
- eidos_strprefix
- eidos_strsplit
- eidos_strsuffix
- eidos_substr
- eidos_apply
- eidos_array
- eidos_cbind
- eidos_diag

- eidos_defineConstant
- eidos_defineGlobal
- eidos_exists
- eidos_functionSignature
- eidos_functionSource
- eidos_getSeed
- eidos_license
- eidos_ls
- eidos_rm
- eidos_sapply
- eidos_setSeed
- eidos_source
- eidos_stop
- eidos_suppressWarnings
- eidos_sysinfo
- eidos_system
- eidos_time
- eidos_usage
- eidos_version

This class has the following properties:

None. This class has no properties.

See Also

Other Eidos: eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(),

eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`early` *SLiM early() callback*

Description

This callback specifies the code should be called early in the simulation cycle. For details on exactly when `first()`, `late()` and `early()` callbacks are run during a simulation see SLiM Manual: page 541 for "WF" models, or SLiM Manual: page 549 for "nonWF" models.

Usage

```
early()
```

Value

None

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other callbacks: first(), fitnessEffect(), fitness(), initialize(), interaction(), late(), mateChoice(), modifyChild(), mutationEffect(), mutation(), recombination(), reproduction(), slim_callbacks(), survival()

Examples

```
slim_block(1, early(), {
  sim.addSubpop("p1", 100)
})
```

---

eidos_abs *Eidos method abs*

Description

Documentation for Eidos function abs, which is a method of Eidos. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_abs(x)
```

Arguments

x	An object of type numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the absolute value of x. If x is integer, the C++ function llabs() is used and an integer vector is returned; if x is float, the C++ function fabs() is used and a float vector is returned.

Value

An object of type numeric.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_acos` *Eidos method acos*

Description

Documentation for Eidos function `acos`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It

will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_acos(x)
```

Arguments

x	An object of type numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the arc cosine of x using the C++ function acos().

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(),

eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_all` *Eidos method all*

Description

Documentation for Eidos function `all`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_all(x, ...)
```

Arguments

`x`	An object of type logical. See details for description.
`...`	An object of type NA. NA See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns T if all values are T in x and in any other arguments supplied; if any value is F, returns F. All arguments must be of logical type. If all arguments are zero-length, T is returned.

Value

An object of type logical. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(),

eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_any` *Eidos method any*

Description

Documentation for Eidos function `any`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_any(x, ...)
```

Arguments

`x`	An object of type logical. See details for description.
`...`	An object of type NA. NA See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns T if any value is T in x or in any other arguments supplied; if all values are F, returns F. All arguments must be of logical type. If all arguments are zero-length, F is returned.

Value

An object of type logical. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_apply` *Eidos method apply*

Description

Documentation for Eidos function `apply`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_apply(x, margin, lambdaSource)
```

Arguments

`x`	An object of type any or integer or string. See details for description.
`margin`	An object of type any or integer or string. See details for description.
`lambdaSource`	An object of type any or integer or string. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Prior to Eidos 1.6 / SLiM 2.6, sapply() was named apply(), and this function did not yet exist Applies a block of Eidos code to margins of x. This function is essentially an extension of sapply() for use with matrices and arrays; it is recommended that you fully understand sapply() before tackling this function. As with sapply(), the lambda specified by lambdaSource will be executed for subsets of x, and the results will be concatenated together with type-promotion in the style of c() to produce a result. Unlike sapply(), however, the subsets of x used might be rows, columns, or higher-dimensional slices of x, rather than just single elements, depending upon the value of margin. For apply(), x must be a matrix or array (see section 2.9). The apply() function in Eidos is patterned directly after the apply() function in R, and should behave identically, except that dimension indices in Eidos are zero-based whereas in R they are one-based. The margin parameter gives the indices of dimensions of x that will be iterated over when assembling values to supply to lambdaSource. If x is a matrix it has two dimensions: rows, of dimension index 0, and columns, of dimension index 1. These are the indices of the dimension sizes returned by dim(); dim(x)[0] gives the number of rows of x, and dim(x)[1] gives the number of columns. These dimension indices are also apparent when subsetting x; a subset index in position 0, such as x[m,], gives row m of x, whereas a subset index in position 1, such as x[,n], gives column n of x. In the same manner, supplying 0 for margin specifies that subsets of x from x[0,] to x[m,] should be ”passed” to lambdaSource, through the applyValue ”parameter”; dimension 0 is iterated over, whereas dimension 1 is taken in aggregate since it is not included in margin. The final effect of this is that whole rows of x are passed to lambdaSource through applyValue. Similarly, margin=1 would specify that subsets of x from x[,0] to x[,n] should be passed to lambdaSource, resulting in whole 70 columns being passed. Specifying margin=c(0,1) would indicate that dimensions 0 and 1 should both be iterated over (dimension 0 more rapidly), so for a matrix each each individual value of x would be passed to lambdaSource. Specifying margin=c(1,0) would similarly iterate over both dimensions, but dimension 1 more rapidly; the traversal order would therefore be different, and the dimensionality of the result would also differ (see below). For higher-dimensional arrays dimension indices beyond 1 exist, and so margin=c(0,1) or margin=c(1,0) would provide slices of x to lambdaSource, each slice having a specific row and column index. Slices are generated by subsetting in the same way as operator [], but additionally, redundant dimensions are dropped as by drop(). The return value from apply() is built up from the type-promoted concatenated results, as if by the c() function, from the iterated execution of lambdaSource; the only question is what dimensional structure is imposed upon that vector of values. If the results from lambdaSource are not of a consistent length, or are of length zero, then the concatenated

results are returned as a plain vector. If all results are of length n > 1, the return value is an array of dimensions c(n, dim(x)[margin]); in other words, each n-vector provides the lowest dimension of the result, and the sizes of the marginal dimensions are imposed upon the data above that. If all results are of length n == 1, then if a single margin was specified the result is a vector (of length equal to the size of that marginal dimension), or if more than one margin was specified the result is an array of dimension dim(x)[margin]; in other words, the sizes of the marginal dimensions are imposed upon the data. Since apply() iterates over the marginal dimensions in the same manner, these structures follows the structure of the data. The above explanation may not be entirely clear, so let's look at an example. If x is a matrix with two rows and three columns, such as defined by x = matrix(1:6, nrow=2);, then executing apply(x, 0, "sum(applyValue);"); would cause each row of x to be supplied to the lambda through applyValue, and the values in each row would thus be summed to produce 9 12 as a result. The call apply(x, 1, "sum(applyValue);"); would instead sum columns of x, producing 3 7 11 as a result. Now consider using range() rather than sum() in the lambda, thus producing two values for each row or column. The call apply(x, 0, "range(applyValue);"); produces a result of matrix(c(1,5,2,6), nrow=2), with the range of the first row of x, 1-5, in the first column of the result, and the range of the second row of x, 2-6, in the second column. Although visualization becomes more difficult, these same patterns extend to higher dimensions and arbitrary margins of x.

Value

An object of type any.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(),

eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_array` *Eidos method array*

Description

Documentation for Eidos function `array`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_array(data, dim)
```

Arguments

`data`	An object of type any or integer. See details for description.
`dim`	An object of type any or integer. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Creates a new array from the data specified by data, with the dimension sizes specified by dim. The first dimension size in dim is the number of rows, and the second is the number of columns; further entries specify the sizes of higher-order dimensions. As many dimensions may be specified as desired, but with a minimum of two dimensions. An array with two dimensions is a matrix (by definition); note that matrix() may provide a more convenient way to make a new matrix. Each dimension must be of size 1 or greater; 0-size dimensions are not allowed. The elements of data are used to populate the new array; the size of data must therefore be equal to the size of the new array, which is the product of all the values in dim. The new array will be filled in dimension order: one element in each row until a column is filled, then on to the next column in the same manner until all columns are filled, and then onward into the higher-order dimensions in the same manner.

Value

An object of type any.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(),

eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_asFloat` *Eidos method asFloat*

Description

Documentation for Eidos function `asFloat`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_asFloat(x)
```

Arguments

`x`	An object of type any but object. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the conversion to float of x. If x is string and cannot be converted to float, Eidos will throw an error. 67

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_asin` *Eidos method asin*

Description

Documentation for Eidos function `asin`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_asin(x)
```

Arguments

`x`	An object of type numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the arc sine of x using the C++ function asin().

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (`<bhaller@benhaller.com>`) and Philipp W Messer (`<messer@cornell.edu>`)

See Also

Other Eidos: `E`, `eidos_abs()`, `eidos_acos()`, `eidos_all()`, `eidos_any()`, `eidos_apply()`, `eidos_array()`, `eidos_asFloat()`, `eidos_asInteger()`, `eidos_asLogical()`, `eidos_asString()`, `eidos_assert()`, `eidos_atan2()`, `eidos_atan()`, `eidos_beep()`, `eidos_catn()`, `eidos_cat()`, `eidos_cbind()`, `eidos_ceil()`, `eidos_citation()`, `eidos_clock()`, `eidos_cmColors()`, `eidos_color2rgb()`, `eidos_colors()`, `eidos_cor()`, `eidos_cos()`, `eidos_cov()`, `eidos_createDirectory()`, `eidos_cumProduct()`, `eidos_cumSum()`, `eidos_c()`, `eidos_date()`, `eidos_dbeta()`, `eidos_debugIndent()`,

eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_asInteger` *Eidos method asInteger*

Description

Documentation for Eidos function `asInteger`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_asInteger(x)
```

Arguments

`x`	An object of type any but object. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the conversion to integer of x. If x is of type string or float and cannot be converted to integer, Eidos will throw an error.

Value

An object of type integer.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(),

eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_asLogical` *Eidos method asLogical*

Description

Documentation for Eidos function `asLogical`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_asLogical(x)
```

Arguments

`x`	An object of type any but object. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the conversion to logical of x. Recall that in Eidos the empty string "" is considered F, and all other string values are considered T. Converting INF or -INF to logical yields T (since those values are not equal to zero); converting NAN to logical throws an error.

Value

An object of type logical.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_assert` *Eidos method assert*

Description

Documentation for Eidos function `assert`, which is a method of Eidos. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It

will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_assert(assertions, message)
```

Arguments

assertions	An object of type logical. See details for description.
message	An object of type null or string. Must be of length 1 (a singleton). The default value is NULL. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Assert that a condition or conditions are true. If any element of assertions is F, execution will be stopped. A message, "assertion failed", will be printed before stopping; if message is not NULL; its value will then be printed.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(),

eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_asString` *Eidos method asString*

Description

Documentation for Eidos function `asString`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_asString(x)
```

Arguments

`x`	An object of type any but object. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the conversion to string of x. Note that asString(NULL) returns ”NULL” even though NULL is zero-length.

Value

An object of type string.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(),

`eidos_usage()`, `eidos_var()`, `eidos_version()`, `eidos_whichMax()`, `eidos_whichMin()`, `eidos_which()`, `eidos_writeFile()`, `eidos_writeTempFile()`

`eidos_atan` *Eidos method atan*

Description

Documentation for Eidos function `atan`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_atan(x)
```

Arguments

`x`	An object of type numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the arc tangent of x using the C++ function atan().

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (`<bhaller@benhaller.com>`) and Philipp W Messer (`<messer@cornell.edu>`)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

---

eidos_atan2 *Eidos method atan2*

Description

Documentation for Eidos function `atan2`, which is a method of Eidos. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_atan2(x, y)
```

Arguments

x	An object of type numeric or numeric. See details for description.
y	An object of type numeric or numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the arc tangent of y/x using the C++ function atan2(), which uses the signs of both x and y to determine the correct quadrant for the result.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(),

eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_beep` *Eidos method beep*

Description

Documentation for Eidos function `beep`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_beep(soundName)
```

Arguments

`soundName`	An object of type null or string. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Plays a sound or beeps. On macOS in a GUI environment (i.e., in EidosScribe or SLiMgui), the optional parameter soundName can be the name of a sound file to play; in other cases (if soundName is NULL, or at the command line, or on platforms other than OS X) soundName is ignored and a standard system beep is played. When soundName is not NULL, a sound file in a supported format (such as .aiff or .mp3) is searched for sequentially in four standard locations, in this order: ~/Library/Sounds, /Library/Sounds,

/ Network/Library/Sounds, and finally /System/Library/Sounds. Standard OS X sounds located in /System/Library/Sounds include "Basso", "Blow", "Bottle", "Frog", "Funk", "Glass", "Hero", "Morse", "Ping", "Pop", "Purr", "Sosumi", "Submarine", and "Tink". Do not include the file extension, such as .aiff or .mp3, in soundName. CAUTION: When not running in EidosScribe or SLiMgui, it is often the case that the only simple means available to play a beep is to send a BEL character (ASCII 7) to the standard output. Unfortunately, when this is the case, it means that (1) no beep will be audible if output is being redirected into a file, and (2) a control character, ^G, will occur in the output at the point when the beep was requested. It is therefore recommended that beep() be used only when doing interactive work in a terminal shell (or in a GUI), not when producing output files. However, this issue is platform-specific; on some platforms beep() may result in a beep, and no emitted ^G, even when output is redirected. When a ^G must be emitted to the standard output to generate the beep, a warning message will also be emitted to make any associated problems easier to diagnose.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(),

eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_c` *Eidos method c*

Description

Documentation for Eidos function `c`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_c(...)
```

Arguments

...	An object of type . See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the concatenation of all of its parameters into a single vector, stripped of all matrix/array dimensions (see rbind() and cbind() for concatenation that does not strip this information). The parameters will be promoted to the highest type represented among them, and that type will be the return type. NULL values are ignored; they have no effect on the result.

Value

An object of type any.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_cat` *Eidos method cat*

Description

Documentation for Eidos function `cat`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_cat(x, sep, error)
```

Arguments

`x`	An object of type any or string or logical. See details for description.
`sep`	An object of type any or string or logical. Must be of length 1 (a singleton). The default value is `" "`. See details for description.
`error`	An object of type any or string or logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Concatenates output to Eidos's output stream, joined together by sep. The value x that is output may be of any type. A newline is not appended to the output, unlike the behavior of print(); if a trailing newline is desired, you can use "\n" (or use the catn() function). Also unlike print(), cat() tends to emit very literal output; print(logical(0)) will emit "logical(0)", for example - showing a semantic interpretation of the value - whereas cat(logical(0)) will emit nothing at all, since there are no elements in the value (it is zero-length). Similarly, print(NULL) will emit "NULL", but cat(NULL) will emit nothing. By default (when error is F), the output is sent to the standard Eidos output stream. When running at the command line, this sends it to stdout; when running in SLiMgui, this sends it to the simulation window's output textview. If error is T, the output is instead sent to the Eidos error stream. When running at the command line, this sends it to stderr; when running in SLiMgui, the output is routed to the simulation's debugging output window.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_catn` *Eidos method catn*

Description

Documentation for Eidos function `catn`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_catn(x, sep, error)
```

Arguments

`x`	An object of type any or string or logical. The default value is `""`. See details for description.
`sep`	An object of type any or string or logical. Must be of length 1 (a singleton). The default value is `" "`. See details for description.
`error`	An object of type any or string or logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Concatenates output (with a trailing newline) to Eidos's output stream, joined together by sep. The behavior of catn() is identical to that of cat(), except that a final newline character is appended to the output for convenience. For catn() a default value of "" is supplied for x, to allow a simple catn() call with no parameters to emit a newline. By default (when error is F), the output is sent to the standard Eidos output stream. When running at the command line, this sends it to stdout; when running in SLiMgui, this sends it to the simulation window's output textview. If error is T, the output is instead sent to the Eidos error stream. When running at the command line, this sends it to stderr; when running in SLiMgui, the output is routed to the simulation's debugging output window. 63

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_cbind` *Eidos method cbind*

Description

Documentation for Eidos function `cbind`, which is a method of Eidos. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It

will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_cbind(...)
```

Arguments

...	An object of type . See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Combines vectors or matrices by column to produce a single matrix. The parameters must be vectors (which are interpreted by cbind() as if they were one-column matrices) or matrices. They must be of the same type, of the same class if they are of type object, and have the same number of rows. If these conditions are met, the result is a single matrix with the parameters joined together, left to right. Parameters may instead be NULL, in which case they are ignored; or if all parameters are NULL, the result is NULL. A sequence of vectors, matrices, and NULLs may thus be concatenated with the NULL values removed, analogous to c(). Calling cbind(x) is an easy way to create a one-column matrix from a vector. 71 To combine vectors or matrices by row instead, see rbind().

Value

An object of type any.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(),

eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_ceil` *Eidos method ceil*

Description

Documentation for Eidos function `ceil`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_ceil(x)
```

Arguments

`x`	An object of type float. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the ceiling of x: the smallest integral value greater than or equal to x. Note that the return value is float even though integral values are guaranteed, because values could be outside of the range representable by integer.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(),

eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_citation` *Eidos method citation*

Description

Documentation for Eidos function `citation`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_citation(void)
```

Arguments

`void`	An object of type . See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Prints citation information for Eidos to Eidos's output stream.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_clock` *Eidos method clock*

Description

Documentation for Eidos function `clock`, which is a method of Eidos. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_clock(type)
```

Arguments

`type`	An object of type string. Must be of length 1 (a singleton). The default value is `"cpu"`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the value of a system clock. If type is ”cpu”, this returns the current value of the CPU usage clock. This is the amount of CPU time used by the current process, in seconds; it is unrelated to the current time of day (for that, see the time() function). This is useful mainly for determining how 79 much processor time a given section of code takes; clock() can be called before and after a block of code, and the end clock minus the start clock gives the elapsed CPU time consumed in the execution of the block of code. See also the timed parameter of executeLambda(), which automates this procedure. Note that if multiple cores are utilized by the process, the CPU usage clock will be the sum of the CPU usage across all cores, and may therefore run faster than the wall clock. If type is ”mono”, this returns the value of the system’s monotonic clock. This represents userperceived (”wall clock”) elapsed time from some arbitrary timebase (which will not change during the execution of the program), but it will not jump if the time zone or the wall clock time are changed for the system. This clock is useful for measuring user-perceived elapsed time, as described above, and may provide a more useful metric for performance than CPU time if multiple cores are being utilized.

Value

An object of type float. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_cmColors(), eidos_color2rgb(),

eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_cmColors` *Eidos method cmColors*

Description

Documentation for Eidos function `cmColors`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_cmColors(n)
```

Arguments

`n`	An object of type integer. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

This method has been deprecated, and may be removed in a future release of Eidos. In SLiM 3.5 and later, use colors(n, "cm") instead. Generate colors in a "cyan-magenta" color palette.

Value

An object of type string.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(),

eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_color2rgb` *Eidos method color2rgb*

Description

Documentation for Eidos function `color2rgb`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_color2rgb(color)
```

Arguments

`color`	An object of type string. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Converts a color string to RGB. The color string specified in color may be either a named color (see chapter 7) or a color in hexadecimal format such as "#007FC0". The equivalent RGB color is returned as a float vector of length three (red, green, blue). Returned RGB values will be in the interval [0, 1]. This function can also be called with a non-singleton vector of color strings in color. In this case, the returned float value will be a matrix of RGB values, with three columns (red, green, blue) and one row per element of color.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_colors` *Eidos method colors*

Description

Documentation for Eidos function `colors`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_colors(x, name)
```

Arguments

`x`	An object of type numeric or string. See details for description.
`name`	An object of type numeric or string. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Generate colors in a standard color palette. If x is a singleton integer, the returned vector will contain x color strings representing x colors equidistant along the named palette, spanning its full extent. Alternatively, if x is a float vector of values in [0,1], the returned vector will contain one color string for each value in x, representing the color at the corresponding fraction along the named palette (values outside [0,1] will be clamped to that range). (Note that the function signature states the type of x as numeric, but in this function the integer and float cases have completely different semantic meanings.) The color palette specified by name may be any of the following color palettes based upon color palettes in R: ”cm” ”heat” ”terrain” It may also be one of the following color palettes based on color palettes in MATLAB (and the Turbo palette from Anton Mikhailov of the Google AI group, based upon the Jet palette provided by MATLAB): ”parula” ”hot” ”jet” ”turbo” ”gray” Finally, it may be one of the following color palettes based upon color palettes in Matplotlib, also available in the viridis R package. These color palettes are designed to be perceptually uniform, changing continuously and linearly. They are also designed to perform well even for users with redgreen colorblindness; the ”cividis” palette, in particular, is designed to look nearly identical to those with and without red-green colorblindness, to be perceptually uniform in both hue and brightness, and to increase linearly in brightness. 77 ”magma” ”inferno” ”plasma” ”viridis” ”cividis” This function replaces the deprecated cmColors(), heatColors(), and terrainColors() functions, and adds several several additional color palettes to Eidos. See rainbow() for another color palette function.

Value

An object of type string.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_cor` *Eidos method cor*

Description

Documentation for Eidos function `cor`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_cor(x, y)
```

Arguments

`x`	An object of type numeric or numeric. See details for description.
`y`	An object of type numeric or numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the sample Pearson's correlation coefficient between x and y, usually denoted r. The sizes of x and y must be identical. If x and y have a size of 0 or 1, the return value will be NULL. At present it is illegal to call cor() with a matrix or array argument, because the desired behavior in that case has not yet been implemented.

Value

An object of type float. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (`<bhaller@benhaller.com>`) and Philipp W Messer (`<messer@cornell.edu>`)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_cos` *Eidos method cos*

Description

Documentation for Eidos function `cos`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_cos(x)
```

Arguments

x	An object of type numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the cosine of x using the C++ function cos(). 52

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(),

eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_cov` *Eidos method cov*

Description

Documentation for Eidos function `cov`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_cov(x, y)
```

Arguments

`x`	An object of type numeric or numeric. See details for description.
`y`	An object of type numeric or numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the corrected sample covariance between x and y. The sizes of x and y must be identical. If x and y have a size of 0 or 1, the return value will be NULL. At present it is illegal to call cov() with a matrix or array argument, because the desired behavior in that case has not yet been implemented.

Value

An object of type float. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_createDirectory`

Eidos method createDirectory

Description

Documentation for Eidos function `createDirectory`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_createDirectory(path)
```

Arguments

`path`	An object of type string. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Creates a new filesystem directory at the path specified by path and returns a logical value indicating if the creation succeeded (T) or failed (F). If the path already exists, createDirectory() will do nothing to the filesystem, will emit a warning, and will return T to indicate success if the existing path is a directory, or F to indicate failure if the existing path is not a directory.

Value

An object of type logical. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_cumProduct` *Eidos method cumProduct*

Description

Documentation for Eidos function `cumProduct`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_cumProduct(x)
```

Arguments

x	An object of type numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the cumulative product of x: a vector of equal length as x, in which the element at index i is equal to the product of the elements of x across the range 0:i. The return type will match the type of x. If x is of type integer, but all of the values of the cumulative product vector cannot be represented in that type, an error condition will result.

Value

An object of type numeric.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(),

eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_cumSum` *Eidos method cumSum*

Description

Documentation for Eidos function `cumSum`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_cumSum(x)
```

Arguments

`x`	An object of type numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the cumulative sum of x: a vector of equal length as x, in which the element at index i is equal to the sum of the elements of x across the range 0:i. The return type will match the type of x. If x is of type integer, but all of the values of the cumulative sum vector cannot be represented in that type, an error condition will result.

Value

An object of type numeric.

Copyright

This is documentation for a function in the SLiM software, and has been reproduced from the official manual, which can be found here: http://benhaller.com/slim/SLiM_Manual.pdf. This documentation is Copyright © 2016-2020 Philipp Messer. All rights reserved. More information about SLiM can be found on the official website: https://messerlab.org/slim/

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(),

eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_date` *Eidos method date*

Description

Documentation for Eidos function `date`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_date(void)
```

Arguments

`void`	An object of type . See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a standard date string for the current date in the local time of the executing machine. The format is in two digits, then year in four digits, zero-padded and separated by dashes) regardless of the localization of the executing machine, for predictability and consistency.

Value

An object of type string. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_dbeta` *Eidos method dbeta*

Description

Documentation for Eidos function `dbeta`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_dbeta(x, alpha, beta)
```

Arguments

x	An object of type float or numeric or numeric. See details for description.
alpha	An object of type float or numeric or numeric. See details for description.
beta	An object of type float or numeric or numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a vector of probability densities for a beta distribution at quantiles x with parameters alpha and beta. The alpha and beta parameters may either be singletons, specifying a single value to be used for all of the draws, or they may be vectors of the same length as x, specifying a value for each density computation. The probability density function is P(s | ,) = [Γ(+)/Γ( )Γ( )]s −1(1−s) −1, where is alpha and is beta. Both parameters must be greater than 0. 58

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(),

eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_debugIndent` *Eidos method debugIndent*

Description

Documentation for Eidos function `debugIndent`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_debugIndent(void)
```

Arguments

`void`	An object of type . See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the indentation string currently being used to start lines in the debugging output stream. In a pure Eidos context this will currently be the empty string, "". In specific

Contexts, such as SLiM, the debugging output stream may be structured with nested indentation, in which case this string will typically be a series of spaces or tabs. To make your debugging output (such as from cat(), catn(), or print() with the error=T optional argument set) line up with other output at the current level of execution nesting, you can start your new lines of output with this string if you wish.

Value

An object of type string. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(),

eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_defineConstant` *Eidos method defineConstant*

Description

Documentation for Eidos function `defineConstant`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_defineConstant(symbol, value)
```

Arguments

`symbol`	An object of type string or any. Must be of length 1 (a singleton). See details for description.
`value`	An object of type string or any. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Defines a new constant with the name symbol and the value specified by value. The name cannot previously be defined in any way (i.e., as either a variable or a constant). The defined constant acts identically to intrinsic Eidos constants such as T, NAN, and PI, and will remain defined for as long as the Eidos context lives even if it is defined inside a block being executed by executeLambda(), apply(), sapply(), or a Context-defined script block. Syntactically, value may be any value at all; semantically, however, if value is of object type then value's class must be under an internal memory-management scheme called "retain-release". Objects that are not under retain-release can cease to exist whenever the Context is finished using them, and thus a defined constant referencing such an object could become invalid, which must be prevented. Objects that are under retain-release will not cease to exist if they are referenced by a global constant; the reference to them from the global constant "retains" them and keeps them in existence. All object classes built into Eidos are under retain-release; see the SLiM manual (section "SLiM scoping rules") for discussion of which SLiM object classes are under retain-release. Section 4.5 of this manual discusses this topic further.

Value

An object of type void.

Copyright

This is documentation for a function in the SLiM software, and has been reproduced from the official manual, which can be found here: http://benhaller.com/slim/SLiM_Manual.pdf. This documentation is Copyright © 2016-2020 Philipp Messer. All rights reserved. More information about SLiM can be found on the official website: https://messerlab.org/slim/

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(),

`eidos_usage()`, `eidos_var()`, `eidos_version()`, `eidos_whichMax()`, `eidos_whichMin()`, `eidos_which()`, `eidos_writeFile()`, `eidos_writeTempFile()`

`eidos_defineGlobal` *Eidos method defineGlobal*

Description

Documentation for Eidos function `defineGlobal`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_defineGlobal(symbol, value)
```

Arguments

`symbol`	An object of type string or any. Must be of length 1 (a singleton). See details for description.
`value`	An object of type string or any. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Defines a new global variable with the name symbol and the value specified by value. The name cannot previously be defined as a constant. The result is similar to a standard variable assignment with operator =, except that the variable is always defined in the global scope (even if the defineGlobal() call is made inside a user-defined function or other locally-scoped block, such as a SLiM event or callback). This means that the variable will remain defined even after the current scope is exited. Note that global variables can be hidden by local variables with the same name; unlike defined constants, such scoped masking is allowed. Syntactically, value may be any value at all; semantically, however, if value is of object type then value's class must be under an internal memory-management scheme called "retain-release". Objects that are not under retain-release can cease to exist whenever the Context is finished using them, and thus a global variable referencing such an object could become invalid, which must be 80 prevented. Objects that are under retain-release will not cease to exist if they are referenced by a global variable; the reference to them from the global variable "retains" them and keeps them in existence. All object classes built into Eidos are under retain-release; see the SLiM manual (section "SLiM scoping rules") for discussion of which SLiM object classes are under retain-release. Section 4.5 of this manual discusses this topic further. (vNlifso)doCall(string$ functionName, ...) Returns the results from a call to a specified function. The function named by the parameter functionName is called, and the remaining parameters to doCall() are forwarded on to that function verbatim. This can be useful for calling one of a set of similar functions,

such as sin(), cos(), etc., to perform a math function determined at runtime, or one of the as...() family of functions to convert to a type determined at runtime. Note that named arguments and default arguments, beyond the functionName argument, are not supported by doCall(); all arguments to the target function must be specified explicitly, without names. (vNlifso)executeLambda(string$ lambdaSource, [ls$ timed = F]) Executes a block of Eidos code defined by lambdaSource. Eidos allows you to execute lambdas: blocks of Eidos code which can be called directly within the same scope as the caller. Eidos lambdas do not take arguments; for this reason, they are not first-class functions. (Since they share the scope of the caller, however, you may effectively pass values in and out of a lambda using variables.) The string argument lambdaSource may contain one or many Eidos statements as a single string value. Lambdas are represented, to the caller, only as the source code string lambdaSource; the executable code is not made available programmatically. If an error occurs during the tokenization, parsing, or execution of the lambda, that error is raised as usual; executing code inside a lambda does not provide any additional protection against exceptions raised. The return value produced by the code in the lambda is returned by executeLambda(). If the optional parameter timed is T, the total (CPU clock) execution time for the lambda will be printed after the lambda has completed (see clock()); if it is F (the default), no timing information will be printed. The timed parameter may also be "cpu" or "mono" to specifically request timing with the CPU clock (which will count the usage across all cores, and may thus run faster than wall clock time if multiple cores are being utilized) or the monotonic clock (which will correspond, more or less, to elapsed wall clock time regardless of multithreading); see the documentation for clock() for further discussion of these timing options. The current implementation of executeLambda() caches a tokenized and parsed version of lambdaSource, so calling executeLambda() repeatedly on a single source string is much more efficient than calling executeLambda() with a newly constructed string each time. If you can use a string literal for lambdaSource, or reuse a constructed source string stored in a variable, that will improve performance considerably.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(),

eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_deleteFile` *Eidos method deleteFile*

Description

Documentation for Eidos function `deleteFile`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_deleteFile(filePath)
```

Arguments

filePath	An object of type string. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Deletes the file specified by filePath and returns a logical value indicating if the deletion succeeded (T) or failed (F).

Value

An object of type logical. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(),

eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_dexp` *Eidos method dexp*

Description

Documentation for Eidos function `dexp`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_dexp(x, mu)
```

Arguments

`x`	An object of type float or numeric. See details for description.
`mu`	An object of type float or numeric. The default value is `1`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a vector of probability densities for an exponential distribution at quantiles x with mean mu (i.e. rate 1/mu). The mu parameter may either be a singleton, specifying a single value to be used for all of the draws, or they may be vectors of the same length as x, specifying a value for each density computation.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_dgamma` *Eidos method dgamma*

Description

Documentation for Eidos function `dgamma`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_dgamma(x, mean, shape)
```

Arguments

`x`	An object of type float or numeric or numeric. See details for description.
`mean`	An object of type float or numeric or numeric. See details for description.
`shape`	An object of type float or numeric or numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a vector of probability densities for a gamma distribution at quantiles x with mean mean and shape parameter shape. The mean and shape parameters may either be singletons, specifying a single value to be used for all of the draws, or they may be vectors of the same length as x, specifying a value for each density computation. The probability density function is P(s | ,) = [Γ( )]−1s −1exp(−s/), where is the shape parameter shape, and the mean of the distribution given by mean is equal to .

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_diag` *Eidos method diag*

Description

Documentation for Eidos function `diag`, which is a method of Eidos. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_diag(x, nrow, ncol)
```

Arguments

`x`	An object of type any. The default value is `1`. See details for description.
`nrow`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`ncol`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the diagonal of x. This function has four distinct usage patterns (matching R). First, if x is a matrix of any type, it returns the diagonal elements of x as a vector; in this case, nrow and ncol must be NULL. Second, if x is 1 (the default) and nrow is non-NULL, it returns an identity matrix with the requested number of rows (and, if ncol is also non-NULL, the requested number of columns, otherwise the matrix will be square). Third, if x is a singleton integer value and nrow and ncol are NULL, it returns a square identity matrix of size x. Fourth, if x is a logical, integer, or float vector of length at least 2, it returns a matrix that uses the values of x as its diagonal (without recycling or truncation, unlike R) and has F, 0, or 0.0 off-diagonal entries as appropriate. Note that using diag(x), without nrow or ncol, can have unexpected effects if x is a vector that could be of length one. Use diag(x, nrow=length(x)) for consistent behavior.

Value

An object of type any.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(),

eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_dim` *Eidos method dim*

Description

Documentation for Eidos function `dim`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_dim(x)
```

Arguments

`x`	An object of type any. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the dimensions of matrix or array x. The first dimension value is the number of rows, the second is the number of columns, and further values indicate the sizes of higher-order dimensions, identically to how dimensions are supplied to array(). NULL is returned if x is not a matrix or array.

Value

An object of type integer.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(),

eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_dmvnorm` *Eidos method dmvnorm*

Description

Documentation for Eidos function `dmvnorm`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_dmvnorm(x, mu, sigma)
```

Arguments

`x`	An object of type float or numeric or numeric. See details for description.
`mu`	An object of type float or numeric or numeric. See details for description.
`sigma`	An object of type float or numeric or numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a vector of probability densities for a k-dimensional multivariate normal distribution with a length k mean vector mu and a k × k variance-covariance matrix sigma. The mu and sigma parameters are used for all densities. The quantile values, x, should be supplied as a matrix with one row per vector of quantile values and k columns (one column per dimension); for convenience, a single quantile may be supplied as a vector rather than a matrix with just one row. The number of dimensions k must be at least two; for k=1, use dnorm(). Cholesky decomposition of the variance-covariance matrix sigma is involved as an internal step, and this requires that sigma be positive-definite; if it is not, an error will result. When more than one density is needed, it is much more efficient to call dmvnorm() once to generate all of the densities, since the Cholesky decomposition of sigma can then be done just once.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(),

eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_dnorm` *Eidos method dnorm*

Description

Documentation for Eidos function `dnorm`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_dnorm(x, mean, sd)
```

Arguments

`x`	An object of type float or numeric or numeric. See details for description.
`mean`	An object of type float or numeric or numeric. The default value is `0`. See details for description.
`sd`	An object of type float or numeric or numeric. The default value is `1`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a vector of probability densities for a normal distribution at quantiles x with mean mean and standard deviation sd. The mean and sd parameters may either be singletons, specifying a single value to be used for all of the densities, or they may be vectors of the same length as x, specifying a value for each density computation.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

---

eidos_drop *Eidos method drop*

Description

Documentation for Eidos function drop, which is a method of Eidos. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It

will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_drop(x)
```

Arguments

x	An object of type any. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the result of dropping redundant dimensions from matrix or array x. Redundant dimensions are those with a size of exactly 1. Non-redundant dimensions are retained. If only one nonredundant dimension is present, the result is a vector; if more than one non-redundant dimension is present, the result will be a matrix or array. If x is not a matrix or array, it is returned unmodified.

Value

An object of type any.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(),

eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_elementType` *Eidos method elementType*

Description

Documentation for Eidos function `elementType`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_elementType(x)
```

Arguments

`x`	An object of type any. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the element type of x, as a string. For the non-object types, the element type is the same as the type: "NULL", "logical", "integer", "float", or "string". For object type,

however, elementType() returns the name of the type of element contained by the object, such as "Species" or "Mutation" in the Context of SLiM. Contrast this with type().

Value

An object of type string. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(),

eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_exists` *Eidos method exists*

Description

Documentation for Eidos function `exists`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_exists(symbol)
```

Arguments

`symbol`	An object of type string. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a logical vector indicating whether symbols exist. If a symbol has been defined as an intrinsic Eidos constant like T, INF, and PI, or as a Context-defined constant like sim in SLiM, or as a user-defined constant using defineConstant(), or as a variable by assignment, this function returns T. Otherwise, the symbol has not been defined, and exists() returns F. This is commonly used to check whether a user-defined constant already exists, with the intention of defining the constant if it has not already been defined. A vector of symbols may be passed, producing a vector of corresponding results.

Value

An object of type logical.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

---

eidos_exp *Eidos method exp*

Description

Documentation for Eidos function exp, which is a method of Eidos. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It

will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_exp(x)
```

Arguments

`x`	An object of type numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the base-e exponential of x, ex, using the C++ function exp(). This may be somewhat faster than E^x for large vectors.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: `E`, `eidos_abs()`, `eidos_acos()`, `eidos_all()`, `eidos_any()`, `eidos_apply()`, `eidos_array()`, `eidos_asFloat()`, `eidos_asInteger()`, `eidos_asLogical()`, `eidos_asString()`, `eidos_asin()`, `eidos_assert()`, `eidos_atan2()`, `eidos_atan()`, `eidos_beep()`, `eidos_catn()`, `eidos_cat()`, `eidos_cbind()`, `eidos_ceil()`, `eidos_citation()`, `eidos_clock()`, `eidos_cmColors()`, `eidos_color2rgb()`, `eidos_colors()`, `eidos_cor()`, `eidos_cos()`, `eidos_cov()`, `eidos_createDirectory()`, `eidos_cumProduct()`, `eidos_cumSum()`, `eidos_c()`, `eidos_date()`, `eidos_dbeta()`, `eidos_debugIndent()`, `eidos_defineConstant()`, `eidos_defineGlobal()`, `eidos_deleteFile()`, `eidos_dexp()`, `eidos_dgamma()`, `eidos_diag()`, `eidos_dim()`, `eidos_dmvnorm()`, `eidos_dnorm()`, `eidos_drop()`, `eidos_elementType()`, `eidos_exists()`, `eidos_fileExists()`, `eidos_filesAtPath()`, `eidos_findInterval()`, `eidos_float()`, `eidos_floor()`, `eidos_flushFile()`, `eidos_format()`, `eidos_functionSignature()`, `eidos_functionSource()`, `eidos_getSeed()`, `eidos_getwd()`, `eidos_heatColors()`, `eidos_hsv2rgb()`, `eidos_identical()`, `eidos_ifelse()`, `eidos_integerDiv()`, `eidos_integerMod()`, `eidos_integer()`, `eidos_isFinite()`, `eidos_isFloat()`, `eidos_isInfinite()`, `eidos_isInteger()`, `eidos_isLogical()`, `eidos_isNAN()`, `eidos_isNULL()`, `eidos_isObject()`,

eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_fileExists` *Eidos method fileExists*

Description

Documentation for Eidos function `fileExists`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_fileExists(filePath)
```

Arguments

`filePath`	An object of type string. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Checks the existence of the file specified by filePath and returns a logical value indicating if it exists (T) or does not exist (F). This also works for directories.

Value

An object of type logical. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(),

eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_filesAtPath` *Eidos method filesAtPath*

Description

Documentation for Eidos function `filesAtPath`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_filesAtPath(path, fullPaths)
```

Arguments

`path`	An object of type string or logical. Must be of length 1 (a singleton). See details for description.
`fullPaths`	An object of type string or logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a string vector containing the names of all files in a directory specified by path. If the optional parameter fullPaths is T, full filesystem paths are returned for each file; if fullPaths is F (the default), then only the filenames relative to the specified directory are returned. This list includes directories (i.e. subfolders), including the "." and ".." directories on Un*x systems. The list also includes invisible files, such as those that begin with a "." on Un*x systems. This function does not descend recursively into subdirectories. If an error occurs during the read, NULL will be returned.

Value

An object of type string.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

---

eidos_findInterval *Eidos method findInterval*

Description

Documentation for Eidos function `findInterval`, which is a method of Eidos. Note that the R function is a stub, it does not do anything in R (except bring up this documentation).

It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_findInterval(x, vec, rightmostClosed, allInside)
```

Arguments

`x`	An object of type numeric or numeric or logical or logical. See details for description.
`vec`	An object of type numeric or numeric or logical or logical. See details for description.
`rightmostClosed`	An object of type numeric or numeric or logical or logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.
`allInside`	An object of type numeric or numeric or logical or logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a vector of interval indices for the values in x within a vector of non-decreasing breakpoints vec. The returned integer vector contains, for each corresponding element of x, the index of the interval in vec within which that element of x is contained. More precisely, if i is the returned integer vector from findInterval(x, v), and N is length(v), then for each index j in x, it will be true that v[i[j]] x[j] < v[i[j]+1], treating v[-1] as -INF and v[N] as INF, assuming that the two flags rightmostClosed and allInside have their default value of F. The effects of the flags will be discussed below. Note that vec must be non-decreasing; in other words, it must be sorted in ascending order, although it may have duplicate values. The returned vector will thus be equal in length to x, and each of its elements will be in the interval [-1, N-1]. The rightmostClosed flag, if T, alters the above behavior to treat the rightmost interval, vec[N-2] .. vec[N-1], as closed. This means that if x[j]==vec[N-1] (i.e., equals max(vec)), the corresponding result i[j] will be N-2 as for all other values in the last interval. The allInside flag, if T, alters the above behavior to coerce returned indices into 0 .. N-2. In other words, -1 is mapped to 0, and N-1 is mapped to N-2.

Value

An object of type integer.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_float` *Eidos method float*

Description

Documentation for Eidos function `float`, which is a method of Eidos. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It

will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_float(length)
```

Arguments

length	An object of type integer. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a new float vector of the length specified by length, filled with 0.0 values. This can be useful for pre-allocating a vector which you then fill with values by subscripting. 61

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(),

eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_floor` *Eidos method floor*

Description

Documentation for Eidos function `floor`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_floor(x)
```

Arguments

`x`	An object of type float. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the floor of x: the largest integral value less than or equal to x. Note that the return value is float even though integral values are guaranteed, because values could be outside of the range representable by integer.

Value

An object of type float.

Copyright

This is documentation for a function in the SLiM software, and has been reproduced from the official manual, which can be found here: http://benhaller.com/slim/SLiM_Manual.pdf. This documentation is Copyright © 2016-2020 Philipp Messer. All rights reserved. More information about SLiM can be found on the official website: https://messerlab.org/slim/

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(),

eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_flushFile` *Eidos method flushFile*

Description

Documentation for Eidos function `flushFile`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_flushFile(filePath)
```

Arguments

`filePath`	An object of type string. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Flushes buffered content to a file specified by filePath. Normally, written data is buffered by writeFile() if the compress option of that function is T, holding the data in memory rather than writing it to disk immediately. This buffering improves both performance and file size; however, 73 sometimes it is desirable to flush the buffered data to disk with flush() so that the filesystem is up to date. Note that flushing after every write is not recommended, since it will lose all of the benefits of buffering. Calling flushFile() for a path that has not been written to, or is not being buffered, will do nothing. If the flush is successful, T will be returned; if not, F will be returned (but at present, an error will result instead).

Value

An object of type logical. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

---

eidos_format *Eidos method format*

Description

Documentation for Eidos function format, which is a method of Eidos. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It

will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_format(format, x)
```

Arguments

`format`	An object of type string or numeric. Must be of length 1 (a singleton). See details for description.
`x`	An object of type string or numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a vector of formatted strings generated from x, based upon the formatting string format. The format parameter may be any string value, but must contain exactly one escape sequence beginning with the This escape sequence specifies how to format a single value from the vector x. The returned vector contains one string value for each element of x; each string value is identical to the string supplied in format, except with a formatted version of the corresponding value from x substituted in place of the escape sequence. The syntax for format is a subset of the standard C/C++ printf()-style format strings (available in many places online, such as http://en.cppreference.com/w/c/io/fprintf). The escape sequence used to format each value of x is composed of several elements: - A the beginning, initiating the escape sequence (if an actual desired, rather than an escape sequence, modify the style of formatting: • - : The value is left-justified with the field (as opposed to the default of right-justification). • + : The sign of the value is always prepended, even if the value is positive (as opposed to the default of appending the sign only if the value is negative). • space : The value is prepended by a space when a sign is not prepended. This is ignored if the + flag is present, since values are then always prepended by a sign. • # : An alternative format is used. For produced. For nonzero. For zeros follow. • 0 : Leading zeros are used to pad the field instead of spaces. This flag is ignored if the left-justification flag, -, is present. It is also ignored for integer values, if a precision is specified. - An optional minimum field width, specified as an integer value. Fields will be padded out to this minimum width. Padding will be done with space characters by default (or with zeros, if the 0 flag is used), on the left by default (or on the right, if the - flag is used). - An optional precision, given as an integer value preceded by a . character. If no integer value follows the . character, a precision of zero will be used. For integer values of x (formatted with extra zeros on the left if necessary), with a default precision of 1. For float values of x formatted with the minimum number of digits that will appear to the right of the decimal point (with extra zeros on the right if necessary), with a default precision of 6. - A format specifier. For integer values, this may be base-10 output; there is no difference between the two), or octal output), letters), or For float values, this may be form [−]ddd.ddd; there is no difference between the two), scientific notation (of the form [−]d.ddde±dd or [−]d.dddE±dd, respectively), or the formatting of depending upon the range of the value. Note that relative to the standard C/C++ printf()-style behavior, there are a few differences: (1) only a single escape sequence may be present in the format

string, (2) the use of * to defer field width and precision values to a passed parameter is not supported, (3) only integer and float values of x are supported, (4) only the modifiers may be supplied, since Eidos does not support different sizes of the integer and float types. Note also that the Eidos conventions of emitting INF and NAN for infinities and Not-A-Number values respectively is not honored by this function; the strings generated for such 64 values are platform-dependent, following the implementation definition of the C++ compiler used to build Eidos, since format() calls through to snprintf() to assemble the final string values. For example, format("A number: a vector with three elements: "A number: -4.10" "A number: +15.38" "A number: +8.00". The precision of .2 results in two digits after the decimal point, the minimum field width of 7 results in padding of the values on the left (with spaces) to a minimum of seven characters, the flag + causes a sign to be shown on positive values as well as negative values, and the format specifier f leads to the float values of x being formatted in base-10 decimal. One string value is produced in the result vector for each value in the parameter x. These values could then be merged into a single string with paste(), for example, or printed with print() or cat().

Value

An object of type string.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(),

eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_functionSignature`

Eidos method functionSignature

Description

Documentation for Eidos function `functionSignature`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_functionSignature(functionName)
```

Arguments

`functionName`	An object of type null or string. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Prints function signatures for all functions (if functionName is NULL, the default), or for the function named by functionName, to Eidos's output stream. See section 2.7.4 for more information.

Value

An object of type void.

Copyright

This is documentation for a function in the SLiM software, and has been reproduced from the official manual, which can be found here: http://benhaller.com/slim/SLiM_Manual.pdf. This documentation is Copyright © 2016-2020 Philipp Messer. All rights reserved. More information about SLiM can be found on the official website: https://messerlab.org/slim/

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(),

eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_functionSource` *Eidos method functionSource*

Description

Documentation for Eidos function `functionSource`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_functionSource(functionName)
```

Arguments

`functionName`	An object of type string. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Prints the Eidos source code for the function specified by functionName, or prints a diagnostic message if the function is implemented in C++ rather than Eidos. 81

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_getSeed` *Eidos method getSeed*

Description

Documentation for Eidos function `getSeed`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_getSeed(void)
```

Arguments

`void`	An object of type . See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the random number seed. This is the last seed value set using setSeed(); if setSeed() has not been called, it will be a seed value chosen based on the process-id and the current time when Eidos was initialized, unless the Context has set a different seed value.

Value

An object of type integer. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(),

eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_getwd` *Eidos method getwd*

Description

Documentation for Eidos function `getwd`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_getwd(void)
```

Arguments

`void`	An object of type . See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Gets the current filesystem working directory. The filesystem working directory is the directory which will be used as a base path for relative filesystem paths. For example, if the working directory is "~/Desktop" (the Desktop subdirectory within the current user's home directory, as represented by ~), then the filename "foo.txt" would correspond to the filesystem path "~/Desktop/foo.txt", and the relative path "bar/baz/" would correspond to the filesystem path "~/Desktop/bar/baz/". Note that the path returned may not be identical to the path previously set with setwd(), if for example symbolic links are involved; but it

ought to refer to the same actual directory in the filesystem. The initial working directory is - as is generally the case on Un*x - simply the directory given to the running Eidos process by its parent process (the operating system, a shell, a job scheduler, a debugger, or whatever the case may be). If you launch Eidos (or SLiM) from the command line in a Un*x shell, it is typically the current directory in that shell. Before relative filesystem paths are used, you may therefore wish check what the initial working directory is on your platform, with getwd(), if you are not sure. Alternatively, you can simply use setwd() to set the working directory to a known path.

Value

An object of type string. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(),

eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_heatColors` *Eidos method heatColors*

Description

Documentation for Eidos function `heatColors`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_heatColors(n)
```

Arguments

`n`	An object of type integer. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

This method has been deprecated, and may be removed in a future release of Eidos. In SLiM 3.5 and later, use colors(n, "heat") instead. Generate colors in a "heat map" color palette.

Value

An object of type string.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_hsv2rgb` *Eidos method hsv2rgb*

Description

Documentation for Eidos function `hsv2rgb`, which is a method of Eidos. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It

will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_hsv2rgb(hsv)
```

Arguments

hsv	An object of type float. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Converts an HSV color to RGB. The HSV color is specified in hsv as a float vector of length three (hue, saturation, value), and the equivalent RGB color is returned as a float vector of length three (red, green, blue). HSV values will be clamped to the interval [0, 1], and returned RGB values will also be in the interval [0, 1]. This function can also be called with a matrix of HSV values, with three columns (hue, saturation, value). In this case, the returned float value will be a matrix of RGB values, with three columns (red, green, blue) and one row per row of hsv.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(),

eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_identical` *Eidos method identical*

Description

Documentation for Eidos function `identical`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_identical(x, y)
```

Arguments

`x`	An object of type any or any. See details for description.
`y`	An object of type any or any. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a logical value indicating whether two values are identical. If x and y have exactly the same type and size, and all of their corresponding elements are exactly the same, and (for matrices and arrays) their dimensions are identical, this will return T, otherwise it will return F. The test here is for exact equality; an integer value of 1 is not considered identical to a float value of 1.0, for example. Elements in object values must be literally the same element, not simply identical in all of their properties. Type promotion is never done. For testing whether two values are the same, this is generally preferable to the use of operator == or operator !=; see the discussion at section 2.5.1. Note that identical(NULL,NULL) is T.

Value

An object of type logical. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(),

eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_ifelse` *Eidos method ifelse*

Description

Documentation for Eidos function `ifelse`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_ifelse(test, trueValues, falseValues)
```

Arguments

`test`	An object of type logical or any or any. See details for description.
`trueValues`	An object of type logical or any or any. See details for description.
`falseValues`	An object of type logical or any or any. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the result of a vector conditional operation: a vector composed of values from trueValues, for indices where test is T, and values from falseValues, for indices where test is F. The lengths of trueValues and falseValues must either be equal to 1 or to the length of test; however, trueValues and falseValues don't need to be the same length as each other. Furthermore, the type of trueValues and falseValues must be the same (including, if they are object type, their element type). The return will be of the same length as test, and of the same type as trueValues and falseValues. Each element of the return vector will be

taken from the corresponding element of trueValues if the corresponding element of test is T, or from the corresponding element of falseValues if the corresponding element of test is F; if the vector from which the value is to be taken (i.e., trueValues or falseValues) has a length of 1, that single value is used repeatedly, recycling the vector. If test, trueValues, and/or falseValues are matrices or arrays, that will be ignored by ifelse() except that the result will be of the same dimensionality as test. This is quite similar to a function in R of the same name; note, however, that Eidos evaluates all arguments to functions calls immediately, so trueValues and falseValues will be evaluated fully regardless of the values in test, unlike in R. Value expressions without side effects are therefore recommended.

Value

An object of type any.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(),

eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_integer` *Eidos method integer*

Description

Documentation for Eidos function `integer`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_integer(length, fill1, fill2, fill2Indices)
```

Arguments

`length`	An object of type integer. Must be of length 1 (a singleton). See details for description.
`fill1`	An object of type integer. Must be of length 1 (a singleton). The default value is `0`. See details for description.
`fill2`	An object of type integer. Must be of length 1 (a singleton). The default value is `1`. See details for description.
`fill2Indices`	An object of type null or integer. The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a new integer vector of the length specified by length, filled with 0 values by default. This can be useful for pre-allocating a vector which you then fill with values by subscripting. If a value is supplied for fill1, the new vector will be filled with that value instead of the default of 0. Additionally, if a non-NULL vector is supplied for fill2Indices, the indices

specified by fill2Indices will be filled with the value provided by fill2. For example, given the default values of 0 and 1 for fill1 and fill2, the returned vector will contain 1 at all positions specified by fill2Indices, and will contain 0 at all other positions.

Value

An object of type integer.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(),

eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_integerDiv` *Eidos method integerDiv*

Description

Documentation for Eidos function `integerDiv`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_integerDiv(x, y)
```

Arguments

`x`	An object of type integer or integer. See details for description.
`y`	An object of type integer or integer. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the result of integer division of x by y. The / operator in Eidos always produces a float result; if you want an integer result you may use this function instead. If any value of y is 0, an error will result. The parameters x and y must either be of equal length, or one of the two must be a singleton. The precise behavior of integer division, in terms of how rounding and negative values are handled, may be platform dependent; it will be whatever the C++ behavior of integer division is on the given platform. Eidos does not guarantee any particular behavior, so use this function with caution.

Value

An object of type integer.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_integerMod` *Eidos method integerMod*

Description

Documentation for Eidos function `integerMod`, which is a method of Eidos. Note that the R function is a stub, it does not do anything in R (except bring up this documentation).

It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_integerMod(x, y)
```

Arguments

`x`	An object of type integer or integer. See details for description.
`y`	An object of type integer or integer. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the result of integer modulo of x by y. The always produces a float result; if you want an integer result you may use this function instead. If any value of y is 0, an error will result. The parameters x and y must either be of equal length, or one of the two must be a singleton. The precise behavior of integer modulo, in terms of how rounding and negative values are handled, may be platform dependent; it will be whatever the C++ behavior of integer modulo is on the given platform. Eidos does not guarantee any particular behavior, so use this function with caution.

Value

An object of type integer.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: `E`, `eidos_abs()`, `eidos_acos()`, `eidos_all()`, `eidos_any()`, `eidos_apply()`, `eidos_array()`, `eidos_asFloat()`, `eidos_asInteger()`, `eidos_asLogical()`, `eidos_asString()`, `eidos_asin()`, `eidos_assert()`, `eidos_atan2()`, `eidos_atan()`, `eidos_beep()`, `eidos_catn()`, `eidos_cat()`, `eidos_cbind()`, `eidos_ceil()`, `eidos_citation()`, `eidos_clock()`, `eidos_cmColors()`, `eidos_color2rgb()`, `eidos_colors()`, `eidos_cor()`, `eidos_cos()`, `eidos_cov()`, `eidos_createDirectory()`, `eidos_cumProduct()`, `eidos_cumSum()`, `eidos_c()`, `eidos_date()`, `eidos_dbeta()`, `eidos_debugIndent()`, `eidos_defineConstant()`, `eidos_defineGlobal()`, `eidos_deleteFile()`, `eidos_dexp()`, `eidos_dgamma()`, `eidos_diag()`, `eidos_dim()`, `eidos_dmvnorm()`, `eidos_dnorm()`, `eidos_drop()`,

eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_isFinite` *Eidos method isFinite*

Description

Documentation for Eidos function `isFinite`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_isFinite(x)
```

Arguments

`x`	An object of type float. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the finiteness of x: T if x is not INF or NAN, F if x is INF or NAN. INF and NAN are defined only for type float, so x is required to be a float. Note that isFinite() is not the opposite of isInfinite(), because NAN is considered to be neither finite nor infinite.

Value

An object of type logical.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(),

eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_isFloat` *Eidos method isFloat*

Description

Documentation for Eidos function `isFloat`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_isFloat(x)
```

Arguments

`x`	An object of type any. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns T if x is float type, F otherwise.

Value

An object of type logical. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_isInfinite` *Eidos method isInfinite*

Description

Documentation for Eidos function `isInfinite`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_isInfinite(x)
```

Arguments

x	An object of type float. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the infiniteness of x: T if x is INF, F otherwise. INF is defined only for type float, so x is required to be a float. Note that isInfinite() is not the opposite of isFinite(), because NAN is considered to be neither finite nor infinite.

Value

An object of type logical.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(),

eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_isInteger` *Eidos method isInteger*

Description

Documentation for Eidos function `isInteger`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_isInteger(x)
```

Arguments

`x`	An object of type any. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns T if x is integer type, F otherwise.

Value

An object of type logical. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_isLogical` *Eidos method isLogical*

Description

Documentation for Eidos function `isLogical`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_isLogical(x)
```

Arguments

`x`	An object of type any. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns T if x is logical type, F otherwise.

Value

An object of type logical. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (`<bhaller@benhaller.com>`) and Philipp W Messer (`<messer@cornell.edu>`)

See Also

Other Eidos: `E`, `eidos_abs()`, `eidos_acos()`, `eidos_all()`, `eidos_any()`, `eidos_apply()`, `eidos_array()`, `eidos_asFloat()`, `eidos_asInteger()`, `eidos_asLogical()`, `eidos_asString()`, `eidos_asin()`, `eidos_assert()`, `eidos_atan2()`, `eidos_atan()`, `eidos_beep()`, `eidos_catn()`, `eidos_cat()`, `eidos_cbind()`, `eidos_ceil()`, `eidos_citation()`, `eidos_clock()`, `eidos_cmColors()`, `eidos_color2rgb()`, `eidos_colors()`, `eidos_cor()`, `eidos_cos()`, `eidos_cov()`, `eidos_createDirectory()`, `eidos_cumProduct()`, `eidos_cumSum()`, `eidos_c()`, `eidos_date()`, `eidos_dbeta()`, `eidos_debugIndent()`,

eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_isNAN` *Eidos method isNAN*

Description

Documentation for Eidos function `isNAN`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_isNAN(x)
```

Arguments

`x`	An object of type float. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the undefinedness of x: T if x is not NAN, F if x is NAN. NAN is defined only for type float, so x is required to be a float.

Value

An object of type logical.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(),

eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_isNULL` *Eidos method isNULL*

Description

Documentation for Eidos function `isNULL`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_isNULL(x)
```

Arguments

`x`	An object of type any. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns T if x is NULL type, F otherwise.

Value

An object of type logical. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

---

eidos_isObject *Eidos method isObject*

Description

Documentation for Eidos function `isObject`, which is a method of Eidos. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_isObject(x)
```

Arguments

x	An object of type any. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns T if x is object type, F otherwise.

Value

An object of type logical. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(),

eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_isString` *Eidos method isString*

Description

Documentation for Eidos function `isString`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_isString(x)
```

Arguments

`x`	An object of type any. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns T if x is string type, F otherwise.

Value

An object of type logical. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_length` *Eidos method length*

Description

Documentation for Eidos function `length`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_length(x)
```

Arguments

`x`	An object of type any. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the size (e.g., length) of x: the number of elements contained in x. Note that length() is a synonym for size().

Value

An object of type integer. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (`<bhaller@benhaller.com>`) and Philipp W Messer (`<messer@cornell.edu>`)

See Also

Other Eidos: `E`, `eidos_abs()`, `eidos_acos()`, `eidos_all()`, `eidos_any()`, `eidos_apply()`, `eidos_array()`, `eidos_asFloat()`, `eidos_asInteger()`, `eidos_asLogical()`, `eidos_asString()`, `eidos_asin()`, `eidos_assert()`, `eidos_atan2()`, `eidos_atan()`, `eidos_beep()`, `eidos_catn()`, `eidos_cat()`, `eidos_cbind()`, `eidos_ceil()`, `eidos_citation()`, `eidos_clock()`, `eidos_cmColors()`, `eidos_color2rgb()`, `eidos_colors()`, `eidos_cor()`, `eidos_cos()`, `eidos_cov()`, `eidos_createDirectory()`,

eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_license` *Eidos method license*

Description

Documentation for Eidos function `license`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_license(void)
```

Arguments

`void`	An object of type . See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Prints Eidos's license terms to Eidos's output stream.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(),

eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_log` *Eidos method log*

Description

Documentation for Eidos function `log`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_log(x)
```

Arguments

`x`	An object of type numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the base-e logarithm of x using the C++ function log(). 53

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

---

eidos_log10 *Eidos method log10*

Description

Documentation for Eidos function `log10`, which is a method of Eidos. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_log10(x)
```

Arguments

x	An object of type numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the base-10 logarithm of x using the C++ function log10().

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(),

eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_log2` *Eidos method log2*

Description

Documentation for Eidos function `log2`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_log2(x)
```

Arguments

`x`	An object of type numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the base-2 logarithm of x using the C++ function log2().

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_logical` *Eidos method logical*

Description

Documentation for Eidos function `logical`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_logical(length)
```

Arguments

`length`	An object of type integer. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a new logical vector of the length specified by length, filled with F values. This can be useful for pre-allocating a vector which you then fill with values by subscripting.

Value

An object of type logical.

Copyright

Author(s)

Benjamin C Haller (`<bhaller@benhaller.com>`) and Philipp W Messer (`<messer@cornell.edu>`)

See Also

Other Eidos: `E`, `eidos_abs()`, `eidos_acos()`, `eidos_all()`, `eidos_any()`, `eidos_apply()`, `eidos_array()`, `eidos_asFloat()`, `eidos_asInteger()`, `eidos_asLogical()`, `eidos_asString()`, `eidos_asin()`, `eidos_assert()`, `eidos_atan2()`, `eidos_atan()`, `eidos_beep()`, `eidos_catn()`, `eidos_cat()`, `eidos_cbind()`, `eidos_ceil()`, `eidos_citation()`, `eidos_clock()`, `eidos_cmColors()`,

eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_lowerTri` *Eidos method lowerTri*

Description

Documentation for Eidos function `lowerTri`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_lowerTri(x, diag)
```

Arguments

`x`	An object of type any or logical. See details for description.
`diag`	An object of type any or logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the lower triangle of x, which must be a matrix. The return value will be a logical matrix of the same dimensions as x, with elements T in the lower triangle, F elsewhere. If diag is F (the default), the diagonal is not included in the lower triangle; if diag is T, the diagonal is included in the lower triangle (i.e., its elements will be T).

Value

An object of type logical.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(),

eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_ls` *Eidos method ls*

Description

Documentation for Eidos function `ls`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_ls(showSymbolTables)
```

Arguments

`showSymbolTables`
: An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Prints all currently defined variables to Eidos's output stream. See section 2.4.1 for more information. Beginning in Eidos 2.5 (SLiM 3.5), the showSymbolTables optional argument can be set to T to request full information on the current symbol table chain. This will show which symbol table a given symbol is defined in, as well as revealing whether there are other symbols with the same name that have been masked by a local definition. This is mostly useful for debugging.

Value

An object of type void.

Copyright

This is documentation for a function in the SLiM software, and has been reproduced from the official manual, which can be found here: http://benhaller.com/slim/SLiM_Manual.pdf. This documentation is Copyright © 2016-2020 Philipp Messer. All rights reserved. More information about SLiM can be found on the official website: https://messerlab.org/slim/

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(),

eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_match` *Eidos method match*

Description

Documentation for Eidos function `match`, which is a method of Eidos. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_match(x, table)
```

Arguments

`x`	An object of type any or any. See details for description.
`table`	An object of type any or any. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a vector of the positions of (first) matches of x in table. Type promotion is not performed; x and table must be of the same type. For each element of x, the corresponding element in the result will give the position of the first match for that element of x in table; if the element has no match in table, the element in the result vector will be -1. The result is therefore a vector of the same length as x. If a logical result is desired, with T indicating that a match was found for the corresponding element of x, use (match(x, table) >= 0).

Value

An object of type integer.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

---

eidos_matrix *Eidos method matrix*

Description

Documentation for Eidos function `matrix`, which is a method of Eidos. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_matrix(data, nrow, ncol, byrow)
```

Arguments

`data`	An object of type any. See details for description.
`nrow`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`ncol`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`byrow`	An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Creates a new matrix from the data specified by data. By default this creates a one-column matrix. If non-NULL values are supplied for nrow and/or ncol, a matrix will be made with the requested number of rows and/or columns if possible; if the length of data is not compatible with the requested dimensions, an error will result. By default, values from data will populate the matrix by columns, filling each column sequentially before moving on to the next column; if byrow is T the matrix will be populated by rows instead.

Value

An object of type any.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(),

eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_matrixMult` *Eidos method matrixMult*

Description

Documentation for Eidos function `matrixMult`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_matrixMult(x, y)
```

Arguments

`x`	An object of type numeric or numeric. See details for description.
`y`	An object of type numeric or numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the result of matrix multiplication of x with y. In Eidos (as in R), with two matrices A and B the simple product A * B multiplies the corresponding elements of the matrices; in other words, if X is the result of A * B, then Xij = Aij * Bij. This is parallel to the definition of other operators; A + B adds the corresponding elements of the matrices (Xij = Aij + Bij), etc. In R, true matrix multiplication is achieved with a special operator, matrices, and must be conformable according to the standard definition of matrix multiplication (i.e., if x is an n × m matrix then y must be a m × p matrix, and the result will be a n × p matrix). Vectors will not be promoted to matrices by this function, even if such promotion would lead to a conformable matrix.

Value

An object of type numeric.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(),

eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_max` *Eidos method max*

Description

Documentation for Eidos function `max`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_max(x, ...)
```

Arguments

`x`	An object of type any but object. See details for description.
`...`	An object of type NA. NA See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the maximum of x and the other arguments supplied: the single greatest value contained by all of them. All of the arguments must be the same type as x, and the return type will match that of x. If all of the arguments have a size of 0, the return value will be NULL; note that this means that max(x, max(y)) may produce an error, if max(y) is NULL, in cases where max(x, y) does not.

Value

An object of type any but object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_mean` *Eidos method mean*

Description

Documentation for Eidos function `mean`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_mean(x)
```

Arguments

`x`	An object of type logical or integer or float. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the arithmetic mean of x: the sum of x divided by the number of values in x. If x has a size of 0, the return value will be NULL. The unusual parameter type signature lif indicates that x can be logical, integer, or float; if x is logical, it is coerced to integer internally (with F being 0 and T being 1, as always), allowing mean() to calculate the average truth value of a logical vector.

Value

An object of type float. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (`<bhaller@benhaller.com>`) and Philipp W Messer (`<messer@cornell.edu>`)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

---

eidos_min *Eidos method min*

Description

Documentation for Eidos function `min`, which is a method of Eidos. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_min(x, ...)
```

Arguments

`x`	An object of type any but object. See details for description.
`...`	An object of type NA. NA See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the minimum of x and the other arguments supplied: the single smallest value contained by all of them. All of the arguments must be the same type as x, and the return type will match that of x. If all of the arguments have a size of 0, the return value will be NULL; note that this means that min(x, min(y)) may produce an error, if min(y) is NULL, in cases where min(x, y) does not. 56

Value

An object of type any but object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(),

eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_nchar` *Eidos method nchar*

Description

Documentation for Eidos function `nchar`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_nchar(x)
```

Arguments

`x`	An object of type string. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a vector of the number of characters in the string-elements of `x`.

Value

An object of type integer.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_ncol` *Eidos method ncol*

Description

Documentation for Eidos function `ncol`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_ncol(x)
```

Arguments

`x`	An object of type any. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the number of columns in matrix or array x. For vector x, ncol() returns NULL; size() should be used. An equivalent of R's NCOL() function, which treats vectors as 1-column matrices, is not provided but would be trivial to implement as a user-defined function. 72

Value

An object of type integer. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

---

eidos_nrow *Eidos method nrow*

Description

Documentation for Eidos function **nrow**, which is a method of Eidos. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_nrow(x)
```

Arguments

x	An object of type any. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the number of rows in matrix or array x. For vector x, nrow() returns NULL; size() should be used. An equivalent of R's NROW() function, which treats vectors as 1-column matrices, is not provided but would be trivial to implement as a user-defined function.

Value

An object of type integer. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(),

eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_object` *Eidos method object*

Description

Documentation for Eidos function `object`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_object(void)
```

Arguments

`void`	An object of type . See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a new empty object vector. Unlike float(), integer(), logical(), and string(), a length cannot be specified and the new vector contains no elements. This is because there is no default value for the object type. Adding to such a vector is typically done with c(). Note that the return value is of type object<Object>; this method creates an object vector that does not know what element type it contains. Such object vectors may be mixed freely with other object vectors in c() and similar contexts; the result of such mixing will take its object-element type from the object vector with a defined object-element type (if any).

Value

An object of type Object object.

Copyright

This is documentation for a function in the SLiM software, and has been reproduced from the official manual, which can be found here: http://benhaller.com/slim/SLiM_Manual.pdf. This documentation is Copyright © 2016-2020 Philipp Messer. All rights reserved. More information about SLiM can be found on the official website: https://messerlab.org/slim/

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(),

eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_order` *Eidos method order*

Description

Documentation for Eidos function `order`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_order(x, ascending)
```

Arguments

`x`	An object of type any but object or logical. See details for description.
`ascending`	An object of type any but object or logical. Must be of length 1 (a singleton). The default value is `T`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a vector of sorting indices for x: a new integer vector of the same length as x, containing the indices into x that would sort x. In other words, x[order(x)]==sort(x). This can be useful for 65 more complex sorting problems, such as sorting several vectors in parallel by a sort order determined by one of the vectors. If the optional logical parameter ascending is T (the default), then the sorted order will be ascending; if it is F, the sorted order will be descending. The ordering is determined according to the same logic as the < and > operators in Eidos. To easily sort vectors in a single step, use sort() or sortBy(), for non-object and object vectors respectively.

Value

An object of type integer.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

---

eidos_paste *Eidos method paste*

Description

Documentation for Eidos function paste, which is a method of Eidos. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It

will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_paste(..., sep)
```

Arguments

`...`	An object of type NA. NA See details for description.
`sep`	An object of type string. Must be of length 1 (a singleton). The default value is `" "`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a joined string composed from the string representations of the elements of the parameters passed in, taken in order, joined together by sep. Although this function is based upon the R paste() function of the same name, note that it is much simpler and less powerful; in particular, the result is always a singleton string, rather than returning a non-singleton string vector when one of the parameters is a non-singleton. The string representation used by paste() is the same as that emitted by cat().

Value

An object of type string. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: `E`, `eidos_abs()`, `eidos_acos()`, `eidos_all()`, `eidos_any()`, `eidos_apply()`, `eidos_array()`, `eidos_asFloat()`, `eidos_asInteger()`, `eidos_asLogical()`, `eidos_asString()`, `eidos_asin()`, `eidos_assert()`, `eidos_atan2()`, `eidos_atan()`, `eidos_beep()`, `eidos_catn()`, `eidos_cat()`, `eidos_cbind()`, `eidos_ceil()`, `eidos_citation()`, `eidos_clock()`, `eidos_cmColors()`, `eidos_color2rgb()`, `eidos_colors()`, `eidos_cor()`, `eidos_cos()`, `eidos_cov()`, `eidos_createDirectory()`, `eidos_cumProduct()`, `eidos_cumSum()`, `eidos_c()`, `eidos_date()`, `eidos_dbeta()`, `eidos_debugIndent()`, `eidos_defineConstant()`, `eidos_defineGlobal()`, `eidos_deleteFile()`, `eidos_dexp()`, `eidos_dgamma()`, `eidos_diag()`, `eidos_dim()`, `eidos_dmvnorm()`, `eidos_dnorm()`, `eidos_drop()`,

eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_paste0` *Eidos method paste0*

Description

Documentation for Eidos function `paste0`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_paste0(...)
```

Arguments

...	An object of type . See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a joined string composed from the string representations of the elements of the parameters passed in, taken in order, joined together with no separator. This function is identical to paste(), except that no separator is used. Note that this differs from the semantics of paste0() in R.

Value

An object of type string. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(),

eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

---

eidos_pmax *Eidos method pmax*

Description

Documentation for Eidos function pmax, which is a method of Eidos. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_pmax(x, y)
```

Arguments

x	An object of type any but object or any but object. See details for description.
y	An object of type any but object or any but object. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the parallel maximum of x and y: the element-wise maximum for each corresponding pair of elements in x and y. The type of x and y must match, and the returned value will have the same type. In one usage pattern the size of x and y match, in which case the returned value will have the same size. In the other usage pattern either x and y is a singleton, in which case the returned value will match the size of the non-singleton argument, and pairs of elements for comparison will be formed between the singleton's element and each of the elements in the non-singleton.

Value

An object of type any but object.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_pmin` *Eidos method pmin*

Description

Documentation for Eidos function `pmin`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_pmin(x, y)
```

Arguments

`x`	An object of type any but object or any but object. See details for description.
`y`	An object of type any but object or any but object. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the parallel minimum of x and y: the element-wise minimum for each corresponding pair of elements in x and y. The type of x and y must match, and the returned value will have the same type. In one usage pattern the size of x and y match, in which case the returned value will have the same size. In the other usage pattern either x and y is a singleton, in which case the returned value will match the size of the non-singleton argument, and pairs of elements for comparison will be formed between the singleton's element and each of the elements in the non-singleton.

Value

An object of type any but object.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_pnorm` *Eidos method pnorm*

Description

Documentation for Eidos function `pnorm`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_pnorm(q, mean, sd)
```

Arguments

`q`	An object of type float or numeric or numeric. See details for description.
`mean`	An object of type float or numeric or numeric. The default value is `0`. See details for description.
`sd`	An object of type float or numeric or numeric. The default value is `1`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a vector of cumulative distribution function values for a normal distribution at quantiles q with mean mean and standard deviation sd. The mean and sd parameters may either be singletons, specifying a single value to be used for all of the quantiles, or they may be vectors of the same length as q, specifying a value for each quantile.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(),

eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_print` *Eidos method print*

Description

Documentation for Eidos function `print`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_print(x, error)
```

Arguments

`x`	An object of type any or logical. See details for description.
`error`	An object of type any or logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Prints output to Eidos's output stream. The value x that is output may be of any type. A newline is appended to the output. See cat() for a discussion of the differences between print() and cat(). By default (when error is F), the output is sent to the standard Eidos output stream. When running at the command line, this sends it to stdout; when running in SLiMgui, this sends it to the simulation window's output textview. If error is T, the output is instead sent to the Eidos error stream. When running at the command line, this sends it to stderr; when running in SLiMgui, the output is routed to the simulation's debugging output window.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(),

eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_product` *Eidos method product*

Description

Documentation for Eidos function `product`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_product(x)
```

Arguments

`x`	An object of type numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the product of x: the result of multiplying all of the elements of x together. If x is float, the result will be float. If x is integer, things are a bit more complex; the result will be integer if it can fit into the integer type without overflow issues (including during intermediate stages of the computation), otherwise it will be float.

Value

An object of type numeric. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_qnorm` *Eidos method qnorm*

Description

Documentation for Eidos function `qnorm`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_qnorm(p, mean, sd)
```

Arguments

`p`	An object of type float or numeric or numeric. See details for description.
`mean`	An object of type float or numeric or numeric. The default value is `0`. See details for description.
`sd`	An object of type float or numeric or numeric. The default value is `1`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a vector of quantiles for a normal distribution with lower tail probabilities less than p, with mean mean and standard deviation sd. The mean and sd parameters may either be singletons, specifying a single value to be used for all of the quantiles, or they may be vectors of the same length as p, specifying a value for each quantile computation.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (`<bhaller@benhaller.com>`) and Philipp W Messer (`<messer@cornell.edu>`)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_quantile` *Eidos method quantile*

Description

Documentation for Eidos function `quantile`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_quantile(x, probs)
```

Arguments

`x`	An object of type numeric. See details for description.
`probs`	An object of type null or float. The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns sample quantiles of x for the given probabilities. The smallest value in x corresponds to a probability of 0, and the largest value in x to a probability of 1. The probs vector should be a vector of probabilities in [0, 1], or NULL, which is equivalent to c(0.0, 0.25, 0.5, 0.75, 1.0), requesting sample quartiles. The quantile function linearly interpolates between the points of the empirical cumulative distribution function. In other words, if x is a vector of length n+1, then the quantiles with probs equal to (0, 1/n, 2/n, ..., (n−1)/n, 1) are equal to the sorted values of x, and the quantile is a linear function of probs otherwise. Note that there are many ways to compute quantiles; this algorithm corresponds to R's default "type 7" algorithm.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(),

eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_rainbow` *Eidos method rainbow*

Description

Documentation for Eidos function `rainbow`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_rainbow(n, s, v, start, end, ccw)
```

Arguments

`n`	An object of type integer. Must be of length 1 (a singleton). See details for description.
`s`	An object of type float. Must be of length 1 (a singleton). The default value is `1.0`. See details for description.

`v`	An object of type float. Must be of length 1 (a singleton). The default value is `1.0`. See details for description.
`start`	An object of type float. Must be of length 1 (a singleton). The default value is `0.0`. See details for description.
`end`	An object of type null or float. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`ccw`	An object of type logical. Must be of length 1 (a singleton). The default value is `T`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Generate colors in a "rainbow" color palette. The number of colors desired is passed in n, and the returned vector will contain n color strings. Parameters s and v control the saturation and value of the rainbow colors generated. The color sequence begins with the hue start, and ramps to the hue end, in a counter-clockwise direction around the standard HSV color wheel if ccw is T (the default, following R), otherwise in a clockwise direction. If end is NULL (the default), a value of (n-1)/n is used, producing a complete rainbow around the color wheel when start is also the default value of 0.0. See colors() for other color palettes.

Value

An object of type string.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(),

eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_range` *Eidos method range*

Description

Documentation for Eidos function `range`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_range(x, ...)
```

Arguments

`x`	An object of type numeric. See details for description.
`...`	An object of type NA. NA See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the range of x and the other arguments supplied: a vector of length 2 composed of the minimum and maximum values contained by all of them, at indices 0 and 1 respectively. All of the arguments must be the same type as x, and the return type will match that of x. If all of the arguments have a size of 0, the return value will be NULL; note that this means that range(x, range(y)) may produce an error, if range(y) is NULL, in cases where range(x, y) does not.

Value

An object of type numeric.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(),

eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_rank` *Eidos method rank*

Description

Documentation for Eidos function `rank`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_rank(x, tiesMethod)
```

Arguments

`x`	An object of type numeric or string. See details for description.
`tiesMethod`	An object of type numeric or string. Must be of length 1 (a singleton). The default value is `"average"`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the ranks of the elements of x: a vector of length L (the length of x), composed of the relative ranks, from 1 to L, of each corresponding element of x. The tiesMethod parameter may be any of ”average” (the default), ”first”, ”last”, ”max”, or ”min” (”random”, supported by R, is not supported by Eidos at this time but could be added if needed). For ”average”, the return value is of type float; for all others, it is of type integer. (Note that the return type does not depend upon the type of x.) The result for all of these tiesMethod values is identical (except for type) if the elements of x are unique; the difference between these methods is in how ties are resolved. Suppose that n elements of x are tied (because they are equal), corresponding to ranks k through k+n−1. For tiesMethod ”average”, all n tied elements receive the same rank, (k + (n−1)/2), which is the average of the ranks.

For "first", the first tied element receives rank k, upward to the last tied element receiving rank k+n−1. For "last", the last tied element receives rank k, downward to the first tied element receiving rank k+n−1. For "max", all n tied element receive the maximum rank, k+n−1. For "min", all n tied element receive the minimum rank, k.

Value

An object of type numeric.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(),

eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_rbeta` *Eidos method rbeta*

Description

Documentation for Eidos function `rbeta`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_rbeta(n, alpha, beta)
```

Arguments

`n`	An object of type integer or numeric or numeric. Must be of length 1 (a singleton). See details for description.
`alpha`	An object of type integer or numeric or numeric. See details for description.
`beta`	An object of type integer or numeric or numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a vector of n random draws from a beta distribution with parameters alpha and beta. The alpha and beta parameters may either be singletons, specifying a single value to be used for all of the draws, or they may be vectors of length n, specifying a value for each draw. Draws are made from a beta distribution with probability density P(s | ,) = [Γ(+)/Γ( )Γ( )]s −1(1−s) −1, where is alpha and is beta. Both parameters must be greater than 0. The values drawn are in the interval [0, 1]. 59

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_rbind` *Eidos method rbind*

Description

Documentation for Eidos function `rbind`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_rbind(...)
```

Arguments

...	An object of type . See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Combines vectors or matrices by row to produce a single matrix. The parameters must be vectors (which are interpreted by rbind() as if they were one-row matrices) or matrices. They must be of the same type, of the same class if they are of type object, and have the same number of columns. If these conditions are met, the result is a single matrix with the parameters joined together, top to bottom. Parameters may instead be NULL, in which case they are ignored; or if all parameters are NULL, the result is NULL. A sequence of vectors, matrices, and NULLs may thus be concatenated with the NULL values removed, analogous to c(). Calling rbind(x) is an easy way to create a one-row matrix from a vector. To combine vectors or matrices by column instead, see cbind().

Value

An object of type any.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

---

eidos_rbinom *Eidos method rbinom*

Description

Documentation for Eidos function `rbinom`, which is a method of Eidos. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_rbinom(n, size, prob)
```

Arguments

n	An object of type integer or integer or float. Must be of length 1 (a singleton). See details for description.
size	An object of type integer or integer or float. See details for description.
prob	An object of type integer or integer or float. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a vector of n random draws from a binomial distribution with a number of trials specified by size and a probability of success specified by prob. The size and prob parameters may either be singletons, specifying a single value to be used for all of the draws, or they may be vectors of length n, specifying a value for each draw.

Value

An object of type integer.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(),

eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_rcauchy` *Eidos method rcauchy*

Description

Documentation for Eidos function `rcauchy`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_rcauchy(n, location, scale)
```

Arguments

`n`	An object of type integer or numeric or numeric. Must be of length 1 (a singleton). See details for description.
`location`	An object of type integer or numeric or numeric. The default value is `0`. See details for description.
`scale`	An object of type integer or numeric or numeric. The default value is `1`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a vector of n random draws from a Cauchy distribution with location location and scale scale. The location and scale parameters may either be singletons, specifying a single value to be used for all of the draws, or they may be vectors of length n, specifying a value for each draw.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(),

eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_rdunif` *Eidos method rdunif*

Description

Documentation for Eidos function `rdunif`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_rdunif(n, min, max)
```

Arguments

`n`	An object of type integer or integer or integer. Must be of length 1 (a singleton). See details for description.
`min`	An object of type integer or integer or integer. The default value is `0`. See details for description.
`max`	An object of type integer or integer or integer. The default value is `1`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a vector of n random draws from a discrete uniform distribution from min to max, inclusive. The min and max parameters may either be singletons, specifying a single value to be used for all of the draws, or they may be vectors of length n, specifying a value for each draw. See runif() for draws from a continuous uniform distribution.

Value

An object of type integer.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_readCSV` *Eidos method readCSV*

Description

Documentation for Eidos function `readCSV`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_readCSV(filePath, colNames, colTypes, sep, quote, dec, comment)
```

Arguments

`filePath`	An object of type string. Must be of length 1 (a singleton). See details for description.
`colNames`	An object of type logical or string. The default value is `T`. See details for description.
`colTypes`	An object of type null or string. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`sep`	An object of type string. Must be of length 1 (a singleton). The default value is `","`. See details for description.
`quote`	An object of type string. Must be of length 1 (a singleton). The default value is `'"'`. See details for description.
`dec`	An object of type string. Must be of length 1 (a singleton). The default value is `"."`. See details for description.
`comment`	An object of type string. Must be of length 1 (a singleton). The default value is `""`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Reads data from a CSV or other delimited file specified by filePath and returns a DataFrame object containing the data in a tabular form. CSV (comma-separated value) files use a somewhat standard file format in which a table of data is provided, with values within a row separated by commas, while rows in the table are separated by newlines. Software from R to Excel (and Eidos; see the serialize() method of Dictionary) can export data in CSV format. This function can actually also read files that use a delimiter other than commas; TSV (tab-separated value) files are a popular alternative. Since there is substantial variation in the exact file format for CSV files, this documentation will try to specify the precise format expected by this function. Note that CSV files represent values differently that Eidos usually does, and some of the format options allowed by readCSV(), such as

decimal commas, are not otherwise available in Eidos. If colNames is T (the default), the first row of data is taken to be a header, containing the string names of the columns in the data table; those names will be used by the resulting DataFrame. If colNames is F, a header row is not expected and column names are auto-generated as X1, X2, etc. If colNames is a string vector, a header row is not expected and colNames will be used as the column names; if additional columns exist beyond the length of colNames their names will be auto-generated. Duplicate column names will generate a warning and be made unique. If colTypes is NULL (the default), the value type for each column will be guessed from the values it contains, as described below. If colTypes is a singleton string, it should contain single-letter codes indicating the desired type for each column, from left to right. The letters lifs have the same meaning as in Eidos signatures (logical, integer, float, and string); in addition, ? may be used to indicate that the type for that column should be guessed as by default, and _ or - may be used to indicate that that column should be skipped - omitted from the returned DataFrame. Other characters in colTypes will result in an error. If additional columns exist beyond the end of the colTypes string their types will be guessed as by default. The separator between values is supplied by sep; it is a comma by default, but a tab can be used instead by supplying tab ("\t" in Eidos), or another character may also be used. If sep is the empty string "", the separator between values is "whitespace", meaning one or more spaces or tabs. When the separator is whitespace, whitespace at the beginning or the end of a line will be ignored. 74 Similarly, the character used to quote string values is a double quote ('"' in Eidos), by default, but another character may be supplied in quote. When the string delimiter is encountered, all following characters are considered to be part of the string until another string delimiter is encountered, terminating the string; this includes spaces, comment characters, newlines, and everything else. Within a string value, the string delimiter itself is used twice in a row to indicate that the delimiter itself is present within the string; for example, if the string value (shown without the usual surrounding quotes to try to avoid confusion) is she said "hello", and the string delimiter is the double quote as it is by default, then in the CSV file the value would be given as "she said ""hello""". The usual Eidos style of escaping characters using a backslash is not part of the CSV standard followed here. (When a string value is provided without using the string delimiter, all following characters are considered part of the string except a newline, the value separator sep, the quote separator quote, and the comment separator comment; if none of those characters are present in the string value, the quote delimiter may be omitted.) The character used to indicate a decimal delimiter in numbers may be supplied with dec; by default this is "." (and so 10.0 would be ten, written with a decimal point), but "," is common in European data files (and so 10,0 would be ten, written with a decimal comma). Note that dec and sep may not be the same, so that it is unambiguous whether 10,0 is two numbers (10 and 0) or one number (10.0). For this reason, European CSV files that use a decimal comma typically use a semicolon as the value separator, which may be supplied with sep=";" to readCSV(). Finally, the remainder of a line following a comment character will be ignored when the file is read; by default comment is the empty string, "", indicating that comments do not exist at all, but "#" is a popular comment prefix. To translate the CSV data into a DataFrame, it is necessary for Eidos to guess what value type each column is unless a column type is specified by colTypes. Quotes surrounding a value are irrelevant to this guess; for example, 1997 and "1997" are both candidates to be integer values (because some programs generate CSV output in which every value is quoted regardless of type). If every value in a column is either true, false, TRUE, FALSE, T, or F, the column will be taken to be logical. Otherwise, if every value in a column is an integer (here defined as

an optional + or -, followed by nothing but decimal digits 0123456789), the column will be taken to be integer. Otherwise, if every value in a column is a floating-point number (here defined as an optional + or -, followed by decimal digits 0123456789, optionally a decimal separator and then optionally more decimal digits, and ending with an optional exponent like e7, E+05, or e-2), the column will be taken to be float; the special values NAN, INF, INFINITY, -INF, and -INFINITY (not case-sensitive) are also candidates to be float (if the rest of the column is also convertible to float), representing the corresponding float constants. Otherwise, the column will be taken to be string. NULL and NA are not recognized by readCSV() in CSV files and will be read as strings. Every line in a CSV file must contain the same number of values (forming a rectangular data table); missing values are not allowed by readCSV() since there is no way to represent them in DataFrame (since Eidos has no equivalent of R's NA value). Spaces are considered part of a data field and are not trimmed, following the RFC 4180 standard. These choices are an attempt to provide optimal behavior for most clients, but given the lack of any universal standard for CSV files, and the lack of any type information in the CSV format, they will not always work as desired; in such cases, it should be reasonably straightforward to preprocess input files using standard Unix text-processing tools like sed and awk.

Value

An object of type DataFrame object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(),

eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_readFile` *Eidos method readFile*

Description

Documentation for Eidos function `readFile`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_readFile(filePath)
```

Arguments

`filePath`	An object of type string. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Reads in the contents of a file specified by filePath and returns a string vector containing the lines (separated by \n and \r characters) of the file. Reading files other than text files is not presently supported. If an error occurs during the read, NULL will be returned.

Value

An object of type string.

Copyright

This is documentation for a function in the SLiM software, and has been reproduced from the official manual, which can be found here: http://benhaller.com/slim/SLiM_Manual.pdf. This documentation is Copyright © 2016-2020 Philipp Messer. All rights reserved. More information about SLiM can be found on the official website: https://messerlab.org/slim/

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(),

eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_rep` *Eidos method rep*

Description

Documentation for Eidos function `rep`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_rep(x, count)
```

Arguments

`x`	An object of type any or integer. See details for description.
`count`	An object of type any or integer. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the repetition of x: the entirety of x is repeated count times. The return type matches the type of x.

Value

An object of type any.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

---

eidos_repEach *Eidos method repEach*

Description

Documentation for Eidos function `repEach`, which is a method of Eidos. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_repEach(x, count)
```

Arguments

`x`	An object of type any or integer. See details for description.
`count`	An object of type any or integer. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the repetition of elements of x: each element of x is repeated. If count is a singleton, it specifies the number of times that each element of x will be repeated. Otherwise, the length of count must be equal to the length of x; in this case, each element of x is repeated a number of times specified by the corresponding value of count.

Value

An object of type any.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(),

eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_rev` *Eidos method rev*

Description

Documentation for Eidos function `rev`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_rev(x)
```

Arguments

`x`	An object of type any. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the reverse of x: a new vector with the same elements as x, but in the opposite order.

Value

An object of type any.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_rexp` *Eidos method rexp*

Description

Documentation for Eidos function `rexp`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_rexp(n, mu)
```

Arguments

`n`	An object of type integer or numeric. Must be of length 1 (a singleton). See details for description.
`mu`	An object of type integer or numeric. The default value is `1`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a vector of n random draws from an exponential distribution with mean mu (i.e. rate 1/mu). The mu parameter may either be a singleton, specifying a single value to be used for all of the draws, or it may be a vector of length n, specifying a value for each draw.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (`<bhaller@benhaller.com>`) and Philipp W Messer (`<messer@cornell.edu>`)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_rf` *Eidos method rf*

Description

Documentation for Eidos function `rf`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_rf(n, d1, d2)
```

Arguments

`n`	An object of type integer or numeric or numeric. Must be of length 1 (a singleton). See details for description.
`d1`	An object of type integer or numeric or numeric. See details for description.
`d2`	An object of type integer or numeric or numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a vector of n random draws from an F-distribution with degrees of freedom d1 and d2. The d1 and d2 parameters may either be singletons, specifying a single value to be used for all of the draws, or they may be vectors of length n, specifying a value for each draw.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(),

eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_rgamma` *Eidos method rgamma*

Description

Documentation for Eidos function `rgamma`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_rgamma(n, mean, shape)
```

Arguments

`n`	An object of type integer or numeric or numeric. Must be of length 1 (a singleton). See details for description.
`mean`	An object of type integer or numeric or numeric. See details for description.
`shape`	An object of type integer or numeric or numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a vector of n random draws from a gamma distribution with mean mean and shape parameter shape. The mean and shape parameters may either be singletons, specifying a single value to be used for all of the draws, or they may be vectors of length n, specifying a value for each draw. Draws are made from a gamma distribution with probability density P(s | ,) = [Γ( )]−1s −1exp(−s/), where is the shape parameter shape, and the mean of the distribution given by mean is equal to . Values of mean less than zero are allowed, and are equivalent (in principle) to the negation of a draw from a gamma distribution with the same shape parameter and the negation of the mean parameter.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(),

eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_rgb2color` *Eidos method rgb2color*

Description

Documentation for Eidos function `rgb2color`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_rgb2color(rgb)
```

Arguments

`rgb`	An object of type float. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Converts an RGB color to a color string. The RGB color is specified in rgb as a float vector of length three (red, green, blue). The equivalent color string is returned as singleton string specifying the color in the format "#RRGGBB", such as "#007FC0". RGB values will be clamped to the interval [0, 1]. 78 This function can also be called with a matrix of RGB values, with three columns (red, green, blue). In this case, the returned string value will be a vector of color strings, with one element per row of rgb.

Value

An object of type string.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_rgb2hsv` *Eidos method rgb2hsv*

Description

Documentation for Eidos function `rgb2hsv`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_rgb2hsv(rgb)
```

Arguments

`rgb`	An object of type float. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Converts an RGB color to HSV. The RGB color is specified in rgb as a float vector of length three (red, green, blue), and the equivalent HSV color is returned as a float vector of length three (hue, saturation, value). RGB values will be clamped to the interval [0, 1], and returned HSV values will also be in the interval [0, 1]. This function can also be called with a matrix of RGB values, with three columns (red, green, blue). In this case, the returned float value will be a matrix of HSV values, with three columns (hue, saturation, value) and one row per row of rgb.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_rgeom` *Eidos method rgeom*

Description

Documentation for Eidos function `rgeom`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_rgeom(n, p)
```

Arguments

n	An object of type integer or float. Must be of length 1 (a singleton). See details for description.
p	An object of type integer or float. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a vector of n random draws from a geometric distribution with parameter p. The p parameter may either be a singleton, specifying a single value to be used for all of the draws, or it may be a vector of length n, specifying a value for each draw. Eidos follows R in using the geometric distribution with support on the set 0, 1, 2, ..., where the drawn value indicates the number of failures prior to success. There is an alternative definition, based upon the number of trial required to get one success, so beware.

Value

An object of type integer.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(),

eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_rlnorm` *Eidos method rlnorm*

Description

Documentation for Eidos function `rlnorm`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_rlnorm(n, meanlog, sdlog)
```

Arguments

`n`	An object of type integer or numeric or numeric. Must be of length 1 (a singleton). See details for description.
`meanlog`	An object of type integer or numeric or numeric. The default value is `0`. See details for description.
`sdlog`	An object of type integer or numeric or numeric. The default value is `1`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a vector of n random draws from a lognormal distribution with mean meanlog and standard deviation sdlog, specified on the log scale. The meanlog and sdlog parameters may either be singletons, specifying a single value to be used for all of the draws, or they may be vectors of length n, specifying a value for each draw.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(),

eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_rm` *Eidos method rm*

Description

Documentation for Eidos function `rm`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_rm(variableNames)
```

Arguments

`variableNames`	An object of type null or string. The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Removes variables from the Eidos namespace; in other words, it causes the variables to become undefined. Variables are specified by their string name in the variableNames parameter. If the optional variableNames parameter is NULL (the default), all variables will be removed (be careful!). In SLiM 3, there was an optional parameter removeConstants that, if T, allowed you to remove defined constants (and then potentially redefine them to have a different value). The removeConstants parameter was removed in SLiM 4, since the defineGlobal() function now provides the ability to define (and redefine) global variables that are not constant.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_rmvnorm` *Eidos method rmvnorm*

Description

Documentation for Eidos function `rmvnorm`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_rmvnorm(n, mu, sigma)
```

Arguments

`n`	An object of type integer or numeric or numeric. Must be of length 1 (a singleton). See details for description.
`mu`	An object of type integer or numeric or numeric. See details for description.
`sigma`	An object of type integer or numeric or numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a matrix of n random draws from a k-dimensional multivariate normal distribution with a length k mean vector mu and a k × k variance-covariance matrix sigma. The mu and sigma parameters are used for all n draws. The draws are returned as a matrix with n rows (one row per draw) and k 60 columns (one column per dimension). The number of dimensions k must be at least two; for k=1, use rnorm(). Cholesky decomposition of the variance-covariance matrix sigma is involved as an internal step, and this requires that sigma be positive-definite; if it is not, an error will result. When more than one draw is needed, it is much more efficient to call rmvnorm() once to generate all of the draws, since the Cholesky decomposition of sigma can then be done just once.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_rnbinom` *Eidos method rnbinom*

Description

Documentation for Eidos function `rnbinom`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It

will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_rnbinom(n, size, prob)
```

Arguments

`n`	An object of type integer or numeric or float. Must be of length 1 (a singleton). See details for description.
`size`	An object of type integer or numeric or float. See details for description.
`prob`	An object of type integer or numeric or float. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a vector of n random draws from a negative binomial distribution representing the number of failures which occur in a sequence of Bernoulli trials before reaching a target number of successful trials specified by size, given a probability of success specified by prob. The mean of this distribution for size s and prob p is s(1−p)/p, with variance s(1−p)/p2. The size and prob parameters may either be singletons, specifying a single value to be used for all of the draws, or they may be vectors of length n, specifying a value for each draw.

Value

An object of type integer.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: `E`, `eidos_abs()`, `eidos_acos()`, `eidos_all()`, `eidos_any()`, `eidos_apply()`, `eidos_array()`, `eidos_asFloat()`, `eidos_asInteger()`, `eidos_asLogical()`, `eidos_asString()`, `eidos_asin()`, `eidos_assert()`, `eidos_atan2()`, `eidos_atan()`, `eidos_beep()`, `eidos_catn()`, `eidos_cat()`, `eidos_cbind()`, `eidos_ceil()`, `eidos_citation()`, `eidos_clock()`, `eidos_cmColors()`, `eidos_color2rgb()`, `eidos_colors()`, `eidos_cor()`, `eidos_cos()`, `eidos_cov()`, `eidos_createDirectory()`, `eidos_cumProduct()`, `eidos_cumSum()`, `eidos_c()`, `eidos_date()`, `eidos_dbeta()`, `eidos_debugIndent()`, `eidos_defineConstant()`, `eidos_defineGlobal()`, `eidos_deleteFile()`, `eidos_dexp()`,

eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_rnorm` *Eidos method rnorm*

Description

Documentation for Eidos function `rnorm`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_rnorm(n, mean, sd)
```

Arguments

`n`	An object of type integer or numeric or numeric. Must be of length 1 (a singleton). See details for description.

`mean`	An object of type integer or numeric or numeric. The default value is `0`. See details for description.
`sd`	An object of type integer or numeric or numeric. The default value is `1`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a vector of n random draws from a normal distribution with mean mean and standard deviation sd. The mean and sd parameters may either be singletons, specifying a single value to be used for all of the draws, or they may be vectors of length n, specifying a value for each draw.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(),

eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_round` *Eidos method round*

Description

Documentation for Eidos function `round`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_round(x)
```

Arguments

`x`	An object of type float. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the round of x: the integral value nearest to x, rounding half-way cases away from 0 (different from the rounding policy of R, which rounds halfway cases toward the nearest even number). Note that the return value is float even though integral values are guaranteed, because values could be outside of the range representable by integer.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_rpois` *Eidos method rpois*

Description

Documentation for Eidos function `rpois`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_rpois(n, lambda)
```

Arguments

`n`	An object of type integer or numeric. Must be of length 1 (a singleton). See details for description.
`lambda`	An object of type integer or numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a vector of n random draws from a Poisson distribution with parameter lambda (not to be confused with the language concept of a "lambda"; lambda here is just the name of a parameter, because the symbol typically used for the parameter of a Poisson distribution is the Greek letter). The lambda parameter may either be a singleton, specifying a single value to be used for all of the draws, or it may be a vector of length n, specifying a value for each draw.

Value

An object of type integer.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_runif` *Eidos method runif*

Description

Documentation for Eidos function `runif`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_runif(n, min, max)
```

Arguments

`n`	An object of type integer or numeric or numeric. Must be of length 1 (a singleton). See details for description.
`min`	An object of type integer or numeric or numeric. The default value is `0`. See details for description.
`max`	An object of type integer or numeric or numeric. The default value is `1`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a vector of n random draws from a continuous uniform distribution from min to max, inclusive. The min and max parameters may either be singletons, specifying a single value to be used for all of the draws, or they may be vectors of length n, specifying a value for each draw. See rdunif() for draws from a discrete uniform distribution.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(),

eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_rweibull` *Eidos method rweibull*

Description

Documentation for Eidos function `rweibull`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_rweibull(n, lambda, k)
```

Arguments

`n`	An object of type integer or numeric or numeric. Must be of length 1 (a singleton). See details for description.
`lambda`	An object of type integer or numeric or numeric. See details for description.
`k`	An object of type integer or numeric or numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a vector of n random draws from a Weibull distribution with scale parameter lambda and shape parameter k, both greater than zero. The lambda and k parameters may either be singletons, specifying a single value to be used for all of the draws, or they may be vectors of length n, specifying a value for each draw. Draws are made from a Weibull distribution with probability distribution P(s | ,k) = (k / k) sk−1 exp(-(s/)k).

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(),

eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_sample` *Eidos method sample*

Description

Documentation for Eidos function `sample`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_sample(x, size, replace, weights)
```

Arguments

`x`	An object of type any. See details for description.
`size`	An object of type integer. Must be of length 1 (a singleton). See details for description.
`replace`	An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.
`weights`	An object of type null or integer or float. The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a vector of size containing a sample from the elements of x. If replace is T, sampling is conducted with replacement (the same element may be drawn more than once); if it is F (the default), then sampling is done without replacement. A vector of weights may be supplied in the optional parameter weights; if not NULL, it must be equal in size to x, all weights must be non-negative, and the sum of the weights must be greater than 0. If weights is NULL (the default), then equal weights are used for all elements of x. An error

occurs if sample() runs out of viable elements from which to draw; most notably, if sampling is done without replacement then size must be at most equal to the size of x, but if weights of zero are supplied then the restriction on size will be even more stringent. The draws are obtained from the standard Eidos random number generator, which might be shared with the Context.

Value

An object of type any.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(),

eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_sapply` *Eidos method sapply*

Description

Documentation for Eidos function `sapply`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_sapply(x, lambdaSource, simplify)
```

Arguments

`x`	An object of type any or string or string. See details for description.
`lambdaSource`	An object of type any or string or string. Must be of length 1 (a singleton). See details for description.
`simplify`	An object of type any or string or string. Must be of length 1 (a singleton). The default value is `"vector"`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Named apply() prior to Eidos 1.6 / SLiM 2.6 Applies a block of Eidos code to the elements of x. This function is sort of a hybrid between c() and executeLambda(); it might be useful to consult the documentation for both of those functions to better understand what sapply() does. For each element in x, the lambda defined by lambdaSource will be called. For the duration of that callout, a variable named applyValue will be defined to have as its value the element of x currently being processed. The expectation is that the lambda will use applyValue in some way, and will return either NULL or a new value (which need not be a singleton, and need not be of the same type as x). The return value of sapply() is generated by concatenating together all of the individual vectors returned by the lambda, in exactly the same manner as the c() function (including the possibility of type promotion). Since this function can be hard to understand at first, here is an example: sapply(1:10, "if (applyValue 49 81. The sapply() operation begins with the vector 1:10. For each element

of that vector, the lambda is called and applyValue is defined with the element value. In this respect, sapply() is actually very much like a for loop. If applyValue is even (as evaluated by the modulo operator, the if statement is F and so NULL is returned by the lambda; this must be done explicitly, since a void return is not allowed by sapply(). If applyValue is odd, on the other hand, the lambda returns its square (as calculated by the exponential operator, ^). Just as with the c() function, NULL values are dropped during concatenation, so the final result contains only the squares of the odd values. This example illustrates that the lambda can "drop" values by returning NULL, so sapply() can be used to select particular elements of a vector that satisfy some condition, much like the subscript operator, []. The example also illustrates that input and result types do not have to match; the vector passed in is integer, whereas the result vector is float. Beginning in Eidos 1.6, a new optional parameter named simplify allows the result of sapply() to be a matrix or array in certain cases, better organizing the elements of the result. If the simplify 82 parameter is "vector", the concatenated result value is returned as a plain vector in all cases; this is the default behavior, for backward compatibility. Two other possible values for simplify are presently supported. If simplify is "matrix", the concatenated result value will be turned into a matrix with one column for each non-NULL value returned by the lambda, as if the values were joined together with cbind(), as long as all of the lambda's return values are either (a) NULL or (b) the same length as the other non-NULL values returned. If simplify is "match", the concatenated result value will be turned into a vector, matrix, or array that exactly matches the dimensions as x, with a one-toone correspondence between x and the elements of the return value just like a unary operator, as long as all of the lambda's return values are singletons (with no NULL values). Both "matrix" and "match" will raise an error if their preconditions are not met, to avoid unexpected behavior, so care should be taken that the preconditions are always met when these options are used. As with executeLambda(), all defined variables are accessible within the lambda, and changes made to variables inside the lambda will persist beyond the end of the sapply() call; the lambda is executing in the same scope as the rest of your code. The sapply() function can seem daunting at first, but it is an essential tool in the Eidos toolbox. It combines the iteration of a for loop, the ability to select elements like operator [], and the ability to assemble results of mixed type together into a single vector like c(), all with the power of arbitrary Eidos code execution like executeLambda(). It is relatively fast, compared to other ways of achieving similar results such as a for loop that accumulates results with c(). Like executeLambda(), sapply() is most efficient if it is called multiple times with a single string script variable, rather than with a newly constructed string for lambdaSource each time. Prior to Eidos 1.6 (SLiM 2.6), sapply() was instead named apply(); it was renamed to sapply() in order to more closely match the naming of functions in R. This renaming allowed a new apply() function to be added to Eidos that operates on the margins of matrices and arrays, similar to the apply() function of R (see apply(), above).

Value

An object of type any.

Copyright

More information about SLiM can be found on the official website: https://messerlab.org/slim/

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_sd` *Eidos method sd*

Description

Documentation for Eidos function `sd`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_sd(x)
```

Arguments

`x`	An object of type numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the corrected sample standard deviation of x. If x has a size of 0 or 1, the return value will be NULL. 57

Value

An object of type float. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: `E`, `eidos_abs()`, `eidos_acos()`, `eidos_all()`, `eidos_any()`, `eidos_apply()`, `eidos_array()`, `eidos_asFloat()`, `eidos_asInteger()`, `eidos_asLogical()`, `eidos_asString()`, `eidos_asin()`, `eidos_assert()`, `eidos_atan2()`, `eidos_atan()`, `eidos_beep()`, `eidos_catn()`, `eidos_cat()`, `eidos_cbind()`, `eidos_ceil()`, `eidos_citation()`, `eidos_clock()`, `eidos_cmColors()`, `eidos_color2rgb()`, `eidos_colors()`, `eidos_cor()`, `eidos_cos()`, `eidos_cov()`, `eidos_createDirectory()`, `eidos_cumProduct()`, `eidos_cumSum()`, `eidos_c()`, `eidos_date()`, `eidos_dbeta()`, `eidos_debugIndent()`, `eidos_defineConstant()`, `eidos_defineGlobal()`, `eidos_deleteFile()`, `eidos_dexp()`, `eidos_dgamma()`, `eidos_diag()`, `eidos_dim()`, `eidos_dmvnorm()`, `eidos_dnorm()`, `eidos_drop()`, `eidos_elementType()`, `eidos_exists()`, `eidos_exp()`, `eidos_fileExists()`, `eidos_filesAtPath()`, `eidos_findInterval()`, `eidos_float()`, `eidos_floor()`, `eidos_flushFile()`, `eidos_format()`, `eidos_functionSignature()`, `eidos_functionSource()`, `eidos_getSeed()`, `eidos_getwd()`, `eidos_heatColors()`, `eidos_hsv2rgb()`, `eidos_identical()`, `eidos_ifelse()`, `eidos_integerDiv()`,

eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_seq` *Eidos method seq*

Description

Documentation for Eidos function `seq`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_seq(from, to, by, length)
```

Arguments

`from`	An object of type numeric. Must be of length 1 (a singleton). See details for description.
`to`	An object of type numeric. Must be of length 1 (a singleton). See details for description.
`by`	An object of type null or integer or float. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`length`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a sequence, starting at from and proceeding in the direction of to until the next value in the sequence would fall beyond to. If the optional parameters by and length are both NULL (the default), the sequence steps by values of 1 or -1 (as needed to proceed in the direction of to). A different step value may be supplied with by, but must have the necessary sign. Alternatively, a sequence length may be supplied in length, in which case the step magnitude will be chosen to produce a sequence of the requested length (with the necessary sign, as before). If from and to are both integer then the return type will be integer when possible (but this may not be possible, depending upon values supplied for by or length), otherwise it will be float. 62

Value

An object of type numeric.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(),

eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_seqAlong` *Eidos method seqAlong*

Description

Documentation for Eidos function `seqAlong`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_seqAlong(x)
```

Arguments

`x`	An object of type any. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns an index sequence along x, from 0 to size(x) - 1, with a step of 1. This is a convenience function for easily obtaining a set of indices to address or iterate through a vector. Any matrix/array dimension information is ignored; the index sequence is suitable for indexing into x as a vector.

Value

An object of type integer.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_seqLen` *Eidos method seqLen*

Description

Documentation for Eidos function `seqLen`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_seqLen(length)
```

Arguments

`length`	An object of type integer. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns an index sequence of length, from 0 to length - 1, with a step of 1; if length is 0 the sequence will be zero-length. This is a convenience function for easily obtaining a set of indices to address or iterate through a vector. Note that when length is 0, using the sequence operator with 0: (length-1) will produce 0 -1, and calling seq(a, b, length=length) will raise an error, but seqLen(length) will return integer(0), making seqLen() particularly useful for generating a sequence of a given length that might be zero.

Value

An object of type integer.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

---

eidos_setDifference *Eidos method setDifference*

Description

Documentation for Eidos function `setDifference`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_setDifference(x, y)
```

Arguments

x	An object of type any or any. See details for description.
y	An object of type any or any. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the set-theoretic (asymmetric) difference of x and y, denoted x y: a vector containing all elements that are in x but are not in y. Duplicate elements will be stripped out, in the same manner as the unique() function. The order of elements in the returned vector is arbitrary and should not be relied upon. The returned vector will be of the same type as x and y, and x and y must be of the same type. The operation performed by this function can be represented graphically using a Venn diagram, where the left circle is x and the right circle is y:

Value

An object of type any.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(),

eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_setIntersection`

Eidos method setIntersection

Description

Documentation for Eidos function `setIntersection`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_setIntersection(x, y)
```

Arguments

`x`	An object of type any or any. See details for description.
`y`	An object of type any or any. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the set-theoretic intersection of x and y, denoted x y: a vector containing all elements that are in both x and y (but not in only x or y). Duplicate elements will be stripped out, in the same manner as the unique() function. The order of elements in the returned vector is arbitrary and should not be relied upon. The returned vector will be of the same type as x and y, and x and y must be of the same type. The operation performed by this function can be represented graphically using a Venn diagram, where the left circle is x and the right circle is y: 54

Value

An object of type any.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(),

eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_setSeed` *Eidos method setSeed*

Description

Documentation for Eidos function `setSeed`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_setSeed(seed)
```

Arguments

`seed`	An object of type integer. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Set the random number seed. Future random numbers will be based upon the seed value set, and the random number sequence generated from a particular seed value is guaranteed to be reproducible. The last seed set can be recovered with the getSeed() function.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_setSymmetricDifference`

Eidos method setSymmetricDifference

Description

Documentation for Eidos function `setSymmetricDifference`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_setSymmetricDifference(x, y)
```

Arguments

`x`	An object of type any or any. See details for description.
`y`	An object of type any or any. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the set-theoretic symmetric difference of x and y, denoted x Δ y: a vector containing all elements that are in x or y, but not in both. Duplicate elements will be stripped out, in the same manner as the unique() function. The order of elements in the returned vector is arbitrary and should not be relied upon. The returned vector will be of the same type as x and y, and x and y must be of the same type. The operation performed by this function can be represented graphically using a Venn diagram, where the left circle is x and the right circle is y:

Value

An object of type any.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

---

eidos_setUnion *Eidos method setUnion*

Description

Documentation for Eidos function `setUnion`, which is a method of Eidos. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_setUnion(x, y)
```

Arguments

x	An object of type any or any. See details for description.
y	An object of type any or any. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the set-theoretic union of x and y, denoted x y: a vector containing all elements that are in x and/or y. Duplicate elements will be stripped out, in the same manner as the unique() function. This function is therefore roughly equivalent to unique(c(x, y)), but this function will probably be faster. The order of elements in the returned vector is arbitrary and should not be relied upon. The returned vector will be of the same type as x and y, and x and y must be of the same type. The operation performed by this function can be represented graphically using a Venn diagram, where the left circle is x and the right circle is y:

Value

An object of type any.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(),

eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_setwd` *Eidos method setwd*

Description

Documentation for Eidos function `setwd`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_setwd(path)
```

Arguments

`path`	An object of type string. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Sets the current filesystem working directory. The filesystem working directory is the directory which will be used as a base path for relative filesystem paths (see getwd() for further discussion). An error will result if the working directory cannot be set to the given path. 75 The current working directory prior to the change will be returned as an invisible string value; the value returned is identical to the value that would have been returned by getwd(), apart from its invisibility. See getwd() for discussion regarding the initial working directory, before it is set with setwd().

Value

An object of type string. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(),

eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_sin` *Eidos method sin*

Description

Documentation for Eidos function `sin`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_sin(x)
```

Arguments

`x`	An object of type numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the sine of x using the C++ function sin().

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

---

eidos_size *Eidos method size*

Description

Documentation for Eidos function size, which is a method of Eidos. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It

will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_size(x)
```

Arguments

x	An object of type any. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the size of x: the number of elements contained in x. Note that length() is a synonym for size().

Value

An object of type integer. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(),

eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_sort` *Eidos method sort*

Description

Documentation for Eidos function `sort`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_sort(x, ascending)
```

Arguments

`x`	An object of type any but object or logical. See details for description.
`ascending`	An object of type any but object or logical. Must be of length 1 (a singleton). The default value is `T`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a sorted copy of x: a new vector with the same elements as x, but in sorted order. If the optional logical parameter ascending is T (the default), then the sorted order will

be ascending; if it is F, the sorted order will be descending. The ordering is determined according to the same logic as the < and > operators in Eidos. To sort an object vector, use sortBy(). To obtain indices for sorting, use order().

Value

An object of type any but object.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(),

eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_sortBy` *Eidos method sortBy*

Description

Documentation for Eidos function `sortBy`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_sortBy(x, property, ascending)
```

Arguments

`x`	An object of type . See details for description.
`property`	An object of type string. Must be of length 1 (a singleton). See details for description.
`ascending`	An object of type logical. Must be of length 1 (a singleton). The default value is `T`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a sorted copy of x: a new vector with the same elements as x, but in sorted order. If the optional logical parameter ascending is T (the default), then the sorted order will be ascending; if it is F, the sorted order will be descending. The ordering is determined according to the same logic as the < and > operators in Eidos. The property argument gives the name of the property within the elements of x according to which sorting should be done. This must be a simple property name; it cannot be a property path. For example, to sort a Mutation vector by the selection coefficients of the mutations, you would simply pass "selectionCoeff", including the quotes, for property. To sort a non-object vector, use sort(). To obtain indices for sorting, use order().

Value

An object of type .

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_source` *Eidos method source*

Description

Documentation for Eidos function `source`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_source(filePath, chdir)
```

Arguments

`filePath`	An object of type string or logical. Must be of length 1 (a singleton). See details for description.
`chdir`	An object of type string or logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Executes the contents of an Eidos source file found at the filesystem path filePath. This is essentially shorthand for calling readFile(), joining the read lines with newlines to form a single string using paste(), and then passing that string to executeLambda(). The source file must consist of complete Eidos statements. Regardless of what the last executed source line evaluates to, source() has no return value. If no file exists at filePath, an error will be raised. The chdir parameter controls the current working directory in effect while the source file is executed. If chdir is F (the default), the current working directory will remain unchanged. If chdir is T, the current working directory will be temporarily changed to the filesystem path at which the source file is located, and restored after execution of the source file is complete.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_sqrt` *Eidos method sqrt*

Description

Documentation for Eidos function `sqrt`, which is a method of Eidos. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It

will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_sqrt(x)
```

Arguments

x	An object of type numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the square root of x using the C++ function sqrt(). This may be somewhat faster than x^0.5 for large vectors.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(),

eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_stop` *Eidos method stop*

Description

Documentation for Eidos function `stop`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_stop(message)
```

Arguments

`message`	An object of type null or string. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Stops execution of Eidos (and of the Context, such as the running SLiM simulation, if applicable), in the event of an error. If the optional message parameter is not NULL, it will be printed to Eidos's error stream prior to stopping.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(),

eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_str` *Eidos method str*

Description

Documentation for Eidos function `str`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_str(x, error)
```

Arguments

`x`	An object of type any or logical. See details for description.
`error`	An object of type any or logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Prints the structure of x: a summary of its type and the values it contains. If x is an object, note that str() produces different results from the str() method of x; the str() function prints the external 66 structure of x (the fact that it is an object, and the number and type of its elements), whereas the str() method prints the internal structure of x (the external structure of all the properties contained by x). By default (when error is F), the output is sent to the standard Eidos output stream. When running at the command line, this sends it to stdout; when running in SLiMgui, this sends it to the simulation window's output textview. If error is T, the output is instead sent to the Eidos error stream. When running at the command line, this sends it to stderr; when running in SLiMgui, the output is routed to the simulation's debugging output window.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_strcontains` *Eidos method strcontains*

Description

Documentation for Eidos function `strcontains`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation).

It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_strcontains(x, s, pos)
```

Arguments

`x`	An object of type string or string or integer. See details for description.
`s`	An object of type string or string or integer. Must be of length 1 (a singleton). See details for description.
`pos`	An object of type string or string or integer. Must be of length 1 (a singleton). The default value is `0`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the occurrence of a string specified by s in each of the elements of x, starting at position pos. Position 0, the default, is the beginning of x; a position of 0 means the entire string is searched. A starting search position that is at or beyond the end of a given element of x is not an error; it just implies that a match will not be found in that element. The existences of matches are returned as a logical vector; if a match was found in a given element, the corresponding value in the returned vector is T, otherwise it is F. This function is a simplified version of strfind(), which returns the positions of matches. The strprefix() and strsuffix() functions are also related.

Value

An object of type logical.

Copyright

Author(s)

Benjamin C Haller (`<bhaller@benhaller.com>`) and Philipp W Messer (`<messer@cornell.edu>`)

See Also

Other Eidos: `E`, `eidos_abs()`, `eidos_acos()`, `eidos_all()`, `eidos_any()`, `eidos_apply()`, `eidos_array()`, `eidos_asFloat()`, `eidos_asInteger()`, `eidos_asLogical()`, `eidos_asString()`, `eidos_asin()`, `eidos_assert()`, `eidos_atan2()`, `eidos_atan()`, `eidos_beep()`, `eidos_catn()`, `eidos_cat()`, `eidos_cbind()`, `eidos_ceil()`, `eidos_citation()`, `eidos_clock()`, `eidos_cmColors()`,

eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_strfind` *Eidos method strfind*

Description

Documentation for Eidos function **strfind**, which is a method of Eidos. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_strfind(x, s, pos)
```

Arguments

`x`	An object of type string or string or integer. See details for description.
`s`	An object of type string or string or integer. Must be of length 1 (a singleton). See details for description.
`pos`	An object of type string or string or integer. Must be of length 1 (a singleton). The default value is `0`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the first occurrence of a string specified by s in each of the elements of x, starting at position pos. Position 0, the default, is the beginning of x; a position of 0 means the entire string is searched. A starting search position that is at or beyond the end of a given element of x is not an error; it just implies that a match will not be found in that element. The positions of matches are returned as an integer vector; if no match was found in a given element, the corresponding value in the returned vector is -1. The strcontains() function may be used when a logical value (found / not found) is desired. 69

Value

An object of type integer.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(),

eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_string` *Eidos method string*

Description

Documentation for Eidos function `string`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_string(length)
```

Arguments

`length`	An object of type integer. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a new string vector of the length specified by length, filled with "" values. This can be useful for pre-allocating a vector which you then fill with values by subscripting.

Value

An object of type string.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(),

eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

---

eidos_strprefix *Eidos method strprefix*

Description

Documentation for Eidos function strprefix, which is a method of Eidos. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_strprefix(x, s)
```

Arguments

x	An object of type string or string. See details for description.
s	An object of type string or string. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the occurrence of a prefix string specified by s at the beginning of each of the elements of x. The existences of prefixes are returned as a logical vector; if a given element begins with the prefix, the corresponding value in the returned vector is T, otherwise it is F.

Value

An object of type logical.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

---

eidos_strsplit *Eidos method strsplit*

Description

Documentation for Eidos function `strsplit`, which is a method of Eidos. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_strsplit(x, sep)
```

Arguments

x	An object of type string or string. Must be of length 1 (a singleton). See details for description.
sep	An object of type string or string. Must be of length 1 (a singleton). The default value is " ". See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns substrings of x that were separated by the separator string sep. Every substring defined by an occurrence of the separator is included, and thus zero-length substrings may be returned. For example, strsplit(".foo..bar.", ".") returns a string vector containing "", "foo", "", "bar", "". In that example, the empty string between "foo" and "bar" in the returned vector is present because there were two periods between foo and bar in the input string - the empty string is the substring between those two separators. If sep is "", a vector of single characters will be returned, resulting from splitting x at every position. Note that paste() performs the inverse operation of strsplit().

Value

An object of type string.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(),

eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_strsuffix` *Eidos method strsuffix*

Description

Documentation for Eidos function **strsuffix**, which is a method of Eidos. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_strsuffix(x, s)
```

Arguments

x	An object of type string or string. See details for description.
s	An object of type string or string. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the occurrence of a suffix string specified by s at the end of each of the elements of x. The existences of suffixes are returned as a logical vector; if a given element ends with the suffix, the corresponding value in the returned vector is T, otherwise it is F.

Value

An object of type logical.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(),

eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_substr` *Eidos method substr*

Description

Documentation for Eidos function `substr`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_substr(x, first, last)
```

Arguments

`x`	An object of type string. See details for description.
`first`	An object of type integer. See details for description.
`last`	An object of type null or integer. The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns substrings extracted from the elements of x, spanning character position first to character position last (inclusive). Character positions are numbered from 0 to nchar(x)-1. Positions that fall outside of that range are legal; a substring range that encompasses no characters will produce an empty string. If first is greater than last, an empty string will also result. If last is NULL (the default), then the substring will extend to the end of the string. The parameters first and last may either be singletons, specifying a single value to be used for all of the substrings, or they may be vectors of the same length as x, specifying a value for each substring.

Value

An object of type string.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_sum` *Eidos method sum*

Description

Documentation for Eidos function `sum`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_sum(x)
```

Arguments

`x`	An object of type logical or integer or float. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the sum of x: the result of adding all of the elements of x together. The unusual parameter type signature lif indicates that x can be logical, integer, or float. If x is float, the result will be float. If x is logical, the result will be integer (the number of T values in x, since the integer values of T and F are 1 and 0 respectively). If x is integer, things are a bit more complex; in this case, the result will be integer if it can fit into the integer type without overflow issues (including during intermediate stages of the computation), otherwise it will be float. Note that floating-point roundoff issues can cause this function to return inexact results when x is float type; this is rarely an issue, but see the sumExact() function for an alternative. 55

Value

An object of type numeric. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_sumExact` *Eidos method sumExact*

Description

Documentation for Eidos function `sumExact`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_sumExact(x)
```

Arguments

`x`	An object of type float. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the exact sum of x: the exact result of adding all of the elements of x together. Unlike the sum() function, sumExact() accepts only type float, since the sum() function is already exact for other types. When summing floating-point values - particularly values that vary across many orders of magnitude - the precision limits of floating-point numbers can lead to roundoff errors that cause the sum() function to return an inexact result. This function does additional work to ensure that the final result is exact within the possible limits of the float type; some roundoff may still inevitably occur, in other words, but a more exact result could not be represented with a value of type float. The disadvantage of using this function instead of sum() is that it is much slower - about 35 times slower, according to one test on macOS, but that will vary across operating systems and hardware. This function is rarely truly needed, but apart from the performance consequences there is no disadvantage to using it.

Value

An object of type float. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(),

eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_suppressWarnings`

Eidos method suppressWarnings

Description

Documentation for Eidos function `suppressWarnings`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_suppressWarnings(suppress)
```

Arguments

`suppress`	An object of type logical. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Turns suppression of warning messages on or off. The suppress flag indicates whether suppression of warnings should be enabled (T) or disabled (F). The previous warning-suppression value is returned by suppressWarnings(), making it easy to suppress warnings from a given call and then return to the previous suppression state afterwards. It is recommended that warnings be suppressed only around short blocks of code (not all the time), so that unexpected but perhaps important warnings are 83 not missed. And of course warnings are generally emitted for good reasons; before deciding to disregard a given warning, make sure that you understand exactly why it is being issued, and are certain that it does not represent a serious problem.

Value

An object of type logical. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(),

eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_sysinfo` *Eidos method sysinfo*

Description

Documentation for Eidos function `sysinfo`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_sysinfo(key)
```

Arguments

`key`	An object of type string. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns information about the system. The information returned by tempdir() depends upon the value of key, which selects one of the pieces of information listed: key value os the name of the OS; "macOS" or "Windows", or "Unix" for all others sysname the name of the kernel release the operating system (kernel) release version the operating system (kernel) version nodename the name by which the machine is known on the network machine the hardware type; often the CPU type (e.g., "x86_64") The value "unknown" will be returned for a key if the correct value cannot be ascertained. Note that the values of keys that refer to the kernel may not be what you expect; for example, on one particular macOS 10.15.7 system, sysname returns "Darwin", release returns "19.6.0", and

version returns "Darwin Kernel Version 19.6.0: Thu Sep 16 20:58:47 PDT 2021; root:xnu-6153.141.40.1~1/RELEASE_X86_64". Further keys can be added if there is information that would be useful, particularly if a cross-platform way to obtain the information can be found.

Value

An object of type any.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(),

eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_system` *Eidos method system*

Description

Documentation for Eidos function `system`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_system(command, args, input, stderr, wait)
```

Arguments

`command`	An object of type string or string or string or logical or logical. Must be of length 1 (a singleton). See details for description.
`args`	An object of type string or string or string or logical or logical. The default value is `""`. See details for description.
`input`	An object of type string or string or string or logical or logical. The default value is `""`. See details for description.
`stderr`	An object of type string or string or string or logical or logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.
`wait`	An object of type string or string or string or logical or logical. Must be of length 1 (a singleton). The default value is `T`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Runs a Un*x command in a /bin/sh shell with optional arguments and input, and returns the result as a vector of output lines. The args parameter may contain a vector of arguments to command; they will be passed directly to the shell without any quoting, so applying the appropriate quoting as needed by /bin/sh is the caller's responsibility. The arguments are appended to command, separated by spaces, and the result is passed to the shell as a single command string, so arguments may simply be given as part of command instead, if preferred. By default no input is supplied to command; if input is non-empty, however, it

will be written to a temporary file (one line per string element) and the standard input of command will be redirected to that temporary file (using standard /bin/sh redirection with <, appended to the command string passed to the shell). By default, output sent to standard error will not be captured (and thus may end up in the output of the SLiM process, or may be lost); if stderr is T, however, the standard error stream will be redirected into standard out (using standard /bin/sh redirection with 2>&1, appended to the command string passed to the shell). Arbitrary command strings involving multiple commands, pipes, redirection, etc., may be used with system(), but may be incompatible with the way that args, input, and stderr are handled by this function, so in this case supplying the whole command string in command may be the simplest course. You may redirect standard error into standard output yourself in command with 2>&1. Supplying input to a complex command line can often be facilitated by the use of parentheses to create a subshell; for example, system("(wc -l | sed 's/ //g')", input=c('foo', 'bar', 'baz')); will supply the input lines to wc courtesy of the subshell started for the () operator. If this strategy doesn't work for the command line you want to execute, you can always write a temporary file yourself using writeFile() or writeTempFile() and redirect that file to standard input in command with <. If wait is T (the default), system() will wait for the command to finish, and return the output generated as a string vector, as described above. If wait is F, system() will instead append " &" to the end of the command line to request that it be run in the background, and it will not collect and 84 return the output from the command; instead it will return string(0) immediately. If the output from the command is needed, it could be redirected to a file, and that file could be checked periodically in Eidos for some indication that the command had completed; if output is not redirected to a file, it may appear in SLiM's output stream. If the final command line executed by system() ends in " &", the behavior of system() should be just as if wait=T had been supplied, but it is recommended to use wait=T instead to ensure that the command line is correctly assembled.

Value

An object of type string.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(),

eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_t` *Eidos method t*

Description

Documentation for Eidos function `t`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_t(x)
```

Arguments

`x`	An object of type any. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the transpose of x, which must be a matrix. This is the matrix reflected across its diagonal; or alternatively, the matrix with its columns written out instead as rows in the same order.

Value

An object of type any.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(),

eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_tabulate` *Eidos method tabulate*

Description

Documentation for Eidos function `tabulate`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_tabulate(bin, maxbin)
```

Arguments

`bin`	An object of type integer. See details for description.
`maxbin`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns occurrence counts for each non-negative integer in bin. Occurrence counts are tabulated into bins for each value 0:maxbin in bin; values outside that range are ignored. The default value of maxbin, NULL, is equivalent to passing maxbin=max(0, bin); in other words, by default the result vector will be exactly large enough to accommodate counts for every integer in bin. In any case, the result vector will contain maxbin+1 elements (some or all of which might be zero, if the occurrence count of that integer in bin is zero). Note that the semantics of this function differ slightly from the tabulate() function in R, because R is 1-based and Eidos is 0-based.

Value

An object of type integer.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

eidos_tan *Eidos method tan*

Description

Documentation for Eidos function `tan`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_tan(x)
```

Arguments

`x`	An object of type numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the tangent of x using the C++ function tan().

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (`<bhaller@benhaller.com>`) and Philipp W Messer (`<messer@cornell.edu>`)

See Also

Other Eidos: `E`, `eidos_abs()`, `eidos_acos()`, `eidos_all()`, `eidos_any()`, `eidos_apply()`, `eidos_array()`, `eidos_asFloat()`, `eidos_asInteger()`, `eidos_asLogical()`, `eidos_asString()`, `eidos_asin()`, `eidos_assert()`, `eidos_atan2()`, `eidos_atan()`, `eidos_beep()`, `eidos_catn()`, `eidos_cat()`, `eidos_cbind()`, `eidos_ceil()`, `eidos_citation()`, `eidos_clock()`, `eidos_cmColors()`, `eidos_color2rgb()`, `eidos_colors()`, `eidos_cor()`, `eidos_cos()`, `eidos_cov()`, `eidos_createDirectory()`, `eidos_cumProduct()`, `eidos_cumSum()`, `eidos_c()`, `eidos_date()`, `eidos_dbeta()`, `eidos_debugIndent()`,

eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_tempdir` *Eidos method tempdir*

Description

Documentation for Eidos function `tempdir`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_tempdir(void)
```

Arguments

`void`	An object of type . See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a path to a directory appropriate for saving temporary files. The path returned by tempdir() is platform-specific, and is not guaranteed to be the same from one run of SLiM to the next. It is guaranteed to end in a slash, so further path components should be appended without a leading slash. At present, on macOS and Linux systems, the path will be "/tmp/"; this may change in future Eidos versions without warning.

Value

An object of type string. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(),

eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_terrainColors` *Eidos method terrainColors*

Description

Documentation for Eidos function `terrainColors`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_terrainColors(n)
```

Arguments

`n`	An object of type integer. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

This method has been deprecated, and may be removed in a future release of Eidos. In SLiM 3.5 and later, use colors(n, "terrain") instead.

Value

An object of type string.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

---

eidos_time *Eidos method time*

Description

Documentation for Eidos function time, which is a method of Eidos. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It

will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_time(void)
```

Arguments

`void`	An object of type . See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns a standard time string for the current time in the local time of the executing machine. The format is minute in two digits, then seconds in two digits, zeropadded and separated by dashes) regardless of the localization of the executing machine, for predictability and consistency. The 24-hour clock time is used (i.e., no AM/PM).

Value

An object of type string. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: `E`, `eidos_abs()`, `eidos_acos()`, `eidos_all()`, `eidos_any()`, `eidos_apply()`, `eidos_array()`, `eidos_asFloat()`, `eidos_asInteger()`, `eidos_asLogical()`, `eidos_asString()`, `eidos_asin()`, `eidos_assert()`, `eidos_atan2()`, `eidos_atan()`, `eidos_beep()`, `eidos_catn()`, `eidos_cat()`, `eidos_cbind()`, `eidos_ceil()`, `eidos_citation()`, `eidos_clock()`, `eidos_cmColors()`, `eidos_color2rgb()`, `eidos_colors()`, `eidos_cor()`, `eidos_cos()`, `eidos_cov()`, `eidos_createDirectory()`, `eidos_cumProduct()`, `eidos_cumSum()`, `eidos_c()`, `eidos_date()`, `eidos_dbeta()`, `eidos_debugIndent()`, `eidos_defineConstant()`, `eidos_defineGlobal()`, `eidos_deleteFile()`, `eidos_dexp()`, `eidos_dgamma()`, `eidos_diag()`, `eidos_dim()`, `eidos_dmvnorm()`, `eidos_dnorm()`, `eidos_drop()`, `eidos_elementType()`, `eidos_exists()`, `eidos_exp()`, `eidos_fileExists()`, `eidos_filesAtPath()`, `eidos_findInterval()`, `eidos_float()`, `eidos_floor()`, `eidos_flushFile()`, `eidos_format()`, `eidos_functionSignature()`, `eidos_functionSource()`, `eidos_getSeed()`, `eidos_getwd()`, `eidos_heatColors()`, `eidos_hsv2rgb()`, `eidos_identical()`, `eidos_ifelse()`, `eidos_integerDiv()`,

eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_trunc` *Eidos method trunc*

Description

Documentation for Eidos function `trunc`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_trunc(x)
```

Arguments

`x`	An object of type float. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the truncation of x: the integral value nearest to, but no larger in magnitude than, x. Note that the return value is float even though integral values are guaranteed, because values could be outside of the range representable by integer.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(),

eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_ttest` *Eidos method ttest*

Description

Documentation for Eidos function `ttest`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_ttest(x, y, mu)
```

Arguments

`x`	An object of type float. See details for description.
`y`	An object of type null or float. The default value is `NULL`. See details for description.
`mu`	An object of type null or float. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the p-value resulting from running a t-test with the supplied data. Two types of t-tests can be performed. If x and y are supplied (i.e., y is non-NULL), a two-sample unpaired two-sided Welch's ttest is conducted using the samples in x and y, each of which must contain at least two elements. The null hypothesis for this test is that the two samples are drawn from populations with the same mean. Other options, such as pooled-variance t-tests, paired t-tests, and one-sided t-tests, are not presently available. If x and mu are supplied (i.e., mu is non-NULL), a one-sample t-test is conducted in which the null hypothesis is that the sample is drawn from a population with mean mu. Note that the results from this function are substantially different from those produced by R. The Eidos ttest() function uses uncorrected sample statistics, which means they will be biased for small sample sizes, whereas R probably uses corrected, unbiased sample statistics. This is an Eidos bug, and might be fixed if anyone complains. If large sample sizes are used, however, the bias is likely to be small, and uncorrected statistics are simpler and faster to compute.

Value

An object of type float. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_type` *Eidos method type*

Description

Documentation for Eidos function `type`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_type(x)
```

Arguments

`x`	An object of type any. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the type of x, as a string: "NULL", "logical", "integer", "float", "string", or "object". Contrast this with elementType(). (lis)grep(string$ pattern, string x, [logical$ ignoreCase = F], [string$ grammar = "ECMAScript"], [string$ value = "indices"], [logical$ fixed = F], [logical$ invert = F]) Searches for regular expression matches in the string-elements of x. Regular expressions (regexes) express patterns that strings can either match or not match; they are very widely used in programming languages and terminal shells. The topic of regexes is very complex, and a great deal of information about them can be found online, including examples and tutorials; this manual will not attempt to document the topic in detail. The grep() function uses a regex supplied in pattern, looking for matches for the regex in each element of x. If ignoreCase is F (the default), the pattern matching will be case sensitive (i.e., uppercase versus lowercase will matter); if it is T, the pattern matching will be case-insensitive. 68 The grammar parameter determines the regex grammar used to find matches. Several options are available. The default, "ECMAScript", is a straightforward regex grammar, the specification for which can be found at https://www.cplusplus.com/reference/regex/ECMAScript/ among many other links. The "basic" grammar uses POSIX basic regular expressions, often called BRE; this is documented at https://en.wikibooks.org/wiki/Regular_Expressions/POSIX_Basic_Regular_Expressions. The "extended" grammar uses POSIX extended regular expressions, often called ERE; this is documented at https://en.wikibooks.org/wiki/Regular_Expressions/POSIX-Extended_Regular_Expressions. The "awk" grammar is based upon the "extended" grammar, with more escapes for non-printing characters. The "grep" and "egrep" grammars are based upon the "basic" and "extended" grammars, respectively, but also allow newline characters ("\n") to separate alternations. If you are not sure which grammar you want to use, "ECMAScript" is recommended. All of these grammars are implemented internally in Eidos using the C++ <regex> library, so if you need clarification on the details of a grammar, you can search

for related C++ materials online. Information about the matches found is returned in one of four ways. If value is "indices" (the default), an integer vector is returned containing the index in x for each match. If value is "elements", a string vector is returned containing the actual string-elements of x for each match. If value is "matches", a string vector is returned containing only the substring that matched, within each string-element in x that matched (if more than one substring in a given element matched, the first match is returned). Finally, if value is "logical" a logical vector is returned, of the same length as x, containing T where the corresponding element of x matched, or F where it did not match. This function therefore encapsulates the functionality of both the grep() and grepl() functions of R; use value="logical" for functionality like that of R's grepl(). If fixed is F (the default), matching is determined using pattern following the specified regex grammar as described above. If fixed is T, matching is instead determined using pattern as a string value to be matched "as is", rather than as a regular expression; the grammar specified does not matter in this case, but ignoreCase still applies. This could be thought of as another grammar value, really, meaning "no grammar", but it is supplied as a separate flag following R. Finally, if invert if F (the default) matching proceeds as normal for the chosen regex grammar, whereas if invert if T matching is inverted: indices, elements, or logical values are returned for the elements of x that did not match. If invert is T, the value parameter may not be "matches". Note that there is not presently any way to extract subpattern matches, nor is there any way to perform replacements of matches.

Value

An object of type string. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(),

eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_unique` *Eidos method unique*

Description

Documentation for Eidos function `unique`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_unique(x, preserveOrder)
```

Arguments

`x`	An object of type any or logical. See details for description.
`preserveOrder`	An object of type any or logical. Must be of length 1 (a singleton). The default value is `T`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the unique values in x. In other words, for each value k in x that occurs at least once, the vector returned will contain k exactly once. If preserveOrder is T (the default), the order of values in x is preserved, taking the first instance of each value; this is relatively slow, with O(n2) performance. If preserveOrder if F instead, the order of values in x is not preserved, and no particular ordering should be relied upon; this is relatively fast, with O(n log n) performance. This performance difference will only matter for large vectors, however; for most applications the default behavior can be retained whether the order of the result matters or not.

Value

An object of type any.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(),

eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_upperTri` *Eidos method upperTri*

Description

Documentation for Eidos function `upperTri`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_upperTri(x, diag)
```

Arguments

`x`	An object of type any or logical. See details for description.
`diag`	An object of type any or logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the upper triangle of x, which must be a matrix. The return value will be a logical matrix of the same dimensions as x, with elements T in the upper triangle, F elsewhere. If diag is F (the default), the diagonal is not included in the upper triangle; if diag is T, the diagonal is included in the upper triangle (i.e., its elements will be T).

Value

An object of type logical.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_usage` *Eidos method usage*

Description

Documentation for Eidos function `usage`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_usage(type)
```

Arguments

`type`	An object of type logical or string. Must be of length 1 (a singleton). The default value is `"rss"`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the memory usage. This is the amount of memory used by the current process, in MB (megabytes); multiply by 1024*1024 to get the usage in bytes. Memory usage is a surprisingly complex topic. One metric reported by usage() is the resident set size, or RSS, which includes memory usage from shared libraries, but does not include memory that is swapped out or has never been used. For most purposes, RSS is a useful metric of memory usage from a practical perspective. On some platforms (AIX, BSD, Solaris) the memory usage reported may be zero, but it should be correct on both macOS and Linux platforms. On macOS, memory pages that have not been used for a while may get compressed by the kernel to reduce the RSS of the process; the RSS metric reported by usage() will reflect the compressed size of such pages, not their original size, so surprising decreases in memory usage may be observed when the kernel decides to compress some memory pages. The RSS is requested with a type of "rss", which is the default; for historical reasons, it can also be requested with a type of F. Another metric reported by usage() is the peak RSS. This is just the highest RSS value that has ever been recorded by the kernel. It should generally mirror the behavior of RSS, except that it ratchets upward monotonically. The peak RSS is requested with a type of "rss_peak"; for historical reasons, it can also be requested with a type of T. The third metric currently reported by usage() is the virtual memory usage. This is essentially the amount of memory used by pages that have been assigned to the process, whether those pages are resident, compressed, or swapped. It is typically much larger than the RSS, because it includes various types of memory that are not counted in the RSS; indeed, for some system configurations the virtual memory usage can be reported as being the entire memory space of the computer. Whether it is a useful metric will be platform-dependent; caveat emptor. This function can be useful for documenting the memory usage of long runs as they are in progress. In SLiM, the RSS could also be used

to trigger tree-sequence simplification with a call to treeSeqSimplify(), to reduce memory usage when it becomes too large, but keep in mind that the simplification process itself may cause a substantial spike in memory usage, and that page compression and swaps may reduce the RSS even though the memory actually used by tree-sequence recording continues to increase. When running under SLiM, other tools for monitoring memory usage include the slim command-line options -m[em] and -M[emhist], and the usage() and outputUsage() methods of Community; see the SLiM manual for more information.

Value

An object of type float. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(),

eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_var` *Eidos method var*

Description

Documentation for Eidos function `var`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_var(x)
```

Arguments

`x`	An object of type numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the corrected sample variance of x. If x has a size of 0 or 1, the return value will be NULL. This is the square of the standard deviation calculated by sd(). At present it is illegal to call var() with a matrix or array argument, because the desired behavior in that case has not yet been implemented.

Value

An object of type float. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_version` *Eidos method version*

Description

Documentation for Eidos function `version`, which is a method of Eidos. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It

will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_version(print)
```

Arguments

print	An object of type logical. Must be of length 1 (a singleton). The default value is T. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Get Eidos's version. There are two ways to use this function. If print is T, the default, then the version number is printed to the Eidos output stream in a formatted manner, like "Eidos version 2.1". If Eidos is attached to a Context that provides a version number, that is also printed, like "SLiM version 3.1". In this case, the Eidos version number, and the Context version number if available, are returned as an invisible float vector. This is most useful when using Eidos interactively. If print is F, on the other hand, nothing is printed, but the returned float vector of version numbers is not 85 invisible. This is useful for scripts that need to test the Eidos or Context version they are running against. In both cases, in the float version numbers returned, a version like 2.4.2 would be returned as 2.42; this would not scale well to subversions greater than nine, so that will be avoided in our versioning.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(),

eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_which` *Eidos method which*

Description

Documentation for Eidos function `which`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_which(x)
```

Arguments

x	An object of type logical. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the indices of T values in x. In other words, if an index k in x is T, then the vector returned will contain k; if index k in x is F, the vector returned will omit k. One way to look at this is that it converts from a logical subsetting vector to an integer (index-based) subsetting vector, without changing which subset positions would be selected.

Value

An object of type integer.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(),

eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_writeFile(), eidos_writeTempFile()

`eidos_whichMax` *Eidos method whichMax*

Description

Documentation for Eidos function `whichMax`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_whichMax(x)
```

Arguments

`x`	An object of type any but object. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the index of the (first) maximum value in x. In other words, if k is equal to the maximum value in x, then the vector returned will contain the index of the first occurrence of k in x. If the maximum value is unique, the result is the same as (but more efficient than) the expression which(x==max(x)), which returns the indices of all of the occurrences of the maximum value in x.

Value

An object of type integer. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMin(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_whichMin` *Eidos method whichMin*

Description

Documentation for Eidos function `whichMin`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_whichMin(x)
```

Arguments

`x`	An object of type any but object. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Returns the index of the (first) minimum value in x. In other words, if k is equal to the minimum value in x, then the vector returned will contain the index of the first occurrence of k in x. If the minimum value is unique, the result is the same as (but more efficient than) the expression which(x==min(x)), which returns the indices of all of the occurrences of the minimum value in x.

Value

An object of type integer. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (`<bhaller@benhaller.com>`) and Philipp W Messer (`<messer@cornell.edu>`)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_which(), eidos_writeFile(), eidos_writeTempFile()

`eidos_writeFile` *Eidos method writeFile*

Description

Documentation for Eidos function `writeFile`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_writeFile(filePath, contents, append, compress)
```

Arguments

`filePath`	An object of type string or string or logical or logical. Must be of length 1 (a singleton). See details for description.
`contents`	An object of type string or string or logical or logical. See details for description.
`append`	An object of type string or string or logical or logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.
`compress`	An object of type string or string or logical or logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Writes or appends to a file specified by filePath with contents specified by contents, a string vector of lines. If append is T, the write will be appended to the existing file (if any) at filePath; if it is F (the default), then the write will replace an existing file at that path. If the write is successful, T will be returned; if not, F will be returned (but at present, an error will result instead). If compress is T, the contents will be compressed with zlib as they are written, and the standard .gz extension for gzip-compressed files will be appended to the filename in filePath if it is not already present. If the compress option is used in conjunction with append==T, Eidos will buffer data to append and flush it to the file in a delayed fashion (for performance reasons), and so appended data may not be visible in the file until later - potentially not until the process ends (i.e., the end of the SLiM simulation, for example). If that delay if undesirable, buffered data can be explicitly flushed to the filesystem with flushFile(). The compress option was added in Eidos 2.4 (SLiM 3.4). Note that readFile() does not currently support reading in compressed data. Note that newline characters will be added at the ends of the lines in contents. If you do not wish to have newlines added, you should use paste() to assemble the elements of contents together into a singleton string.

Value

An object of type logical. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeTempFile()

`eidos_writeTempFile` *Eidos method writeTempFile*

Description

Documentation for Eidos function `writeTempFile`, which is a method of `Eidos`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
eidos_writeTempFile(prefix, suffix, contents, compress)
```

Arguments

`prefix`	An object of type string or string or string or logical. Must be of length 1 (a singleton). See details for description.
`suffix`	An object of type string or string or string or logical. Must be of length 1 (a singleton). See details for description.
`contents`	An object of type string or string or string or logical. See details for description.
`compress`	An object of type string or string or string or logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page NA.

Writes to a unique temporary file with contents specified by contents, a string vector of lines. The filename used will begin with prefix and end with suffix, and will contain six random characters in between; for example, if prefix is "plot1_" and suffix is ".pdf", the generated filename might look like "plot1_r5Mq0t.pdf". It is legal for prefix, suffix, or both to be the empty string, "", but supplying a file extension is usually advisable at minimum. The file will be created inside the / tmp/ directory of the system, which is provided by Un*x systems as a standard location for temporary files; the /tmp/ directory should not be specified as part of prefix (nor should any other directory information). The filename generated is guaranteed not to already exist in /tmp/. The file is created with Un*x permissions 0600, allowing reading and writing only by the user for security. If the write is successful, the full path to the temporary file will be returned; if not, "" will be returned. If compress is T, the contents will be compressed with zlib as they are written, and the standard .gz extension for gzip-compressed files will be appended to the filename suffix in suffix if it is not already present. The compress option was added in Eidos 2.4 (SLiM 3.4). Note that readFile() does not currently support reading in compressed data. Note that newline characters will be added at the ends of the lines in contents. If you do not wish to have newlines added, you should use paste() to assemble the elements of contents together into a singleton string.

Value

An object of type string. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Eidos: E, eidos_abs(), eidos_acos(), eidos_all(), eidos_any(), eidos_apply(), eidos_array(), eidos_asFloat(), eidos_asInteger(), eidos_asLogical(), eidos_asString(), eidos_asin(), eidos_assert(), eidos_atan2(), eidos_atan(), eidos_beep(), eidos_catn(), eidos_cat(), eidos_cbind(), eidos_ceil(), eidos_citation(), eidos_clock(), eidos_cmColors(), eidos_color2rgb(), eidos_colors(), eidos_cor(), eidos_cos(), eidos_cov(), eidos_createDirectory(), eidos_cumProduct(), eidos_cumSum(), eidos_c(), eidos_date(), eidos_dbeta(), eidos_debugIndent(), eidos_defineConstant(), eidos_defineGlobal(), eidos_deleteFile(), eidos_dexp(), eidos_dgamma(), eidos_diag(), eidos_dim(), eidos_dmvnorm(), eidos_dnorm(), eidos_drop(), eidos_elementType(), eidos_exists(), eidos_exp(), eidos_fileExists(), eidos_filesAtPath(), eidos_findInterval(), eidos_float(), eidos_floor(), eidos_flushFile(), eidos_format(), eidos_functionSignature(), eidos_functionSource(), eidos_getSeed(), eidos_getwd(), eidos_heatColors(), eidos_hsv2rgb(), eidos_identical(), eidos_ifelse(), eidos_integerDiv(), eidos_integerMod(), eidos_integer(), eidos_isFinite(), eidos_isFloat(), eidos_isInfinite(), eidos_isInteger(), eidos_isLogical(), eidos_isNAN(), eidos_isNULL(), eidos_isObject(), eidos_isString(), eidos_length(), eidos_license(), eidos_log10(), eidos_log2(), eidos_logical(), eidos_log(), eidos_lowerTri(), eidos_ls(), eidos_match(), eidos_matrixMult(), eidos_matrix(), eidos_max(), eidos_mean(), eidos_min(), eidos_nchar(), eidos_ncol(), eidos_nrow(), eidos_object(), eidos_order(), eidos_paste0(), eidos_paste(), eidos_pmax(), eidos_pmin(), eidos_pnorm(), eidos_print(), eidos_product(), eidos_qnorm(), eidos_quantile(), eidos_rainbow(), eidos_range(), eidos_rank(), eidos_rbeta(), eidos_rbind(), eidos_rbinom(), eidos_rcauchy(), eidos_rdunif(), eidos_readCSV(), eidos_readFile(), eidos_repEach(), eidos_rep(), eidos_rev(), eidos_rexp(), eidos_rf(), eidos_rgamma(), eidos_rgb2color(), eidos_rgb2hsv(), eidos_rgeom(), eidos_rlnorm(), eidos_rmvnorm(), eidos_rm(), eidos_rnbinom(), eidos_rnorm(), eidos_round(), eidos_rpois(), eidos_runif(), eidos_rweibull(), eidos_sample(), eidos_sapply(), eidos_sd(), eidos_seqAlong(), eidos_seqLen(), eidos_seq(), eidos_setDifference(), eidos_setIntersection(), eidos_setSeed(), eidos_setSymmetricDifference(), eidos_setUnion(), eidos_setwd(), eidos_sin(), eidos_size(), eidos_sortBy(), eidos_sort(), eidos_source(), eidos_sqrt(), eidos_stop(), eidos_strcontains(), eidos_strfind(), eidos_string(), eidos_strprefix(), eidos_strsplit(), eidos_strsuffix(), eidos_str(), eidos_substr(), eidos_sumExact(), eidos_sum(), eidos_suppressWarnings(), eidos_sysinfo(), eidos_system(), eidos_tabulate(), eidos_tan(), eidos_tempdir(), eidos_terrainColors(), eidos_time(), eidos_trunc(), eidos_ttest(), eidos_type(), eidos_t(), eidos_unique(), eidos_upperTri(), eidos_usage(), eidos_var(), eidos_version(), eidos_whichMax(), eidos_whichMin(), eidos_which(), eidos_writeFile()

---

end_gen *Extract or set end generation*

Description

Extract or set end generation

Usage

```
end_gen(x)

end_gen(x) <- value
```

Arguments

`x`	A `slimr_script` object
`value`	A end generation value to replace with.

Examples

```
script <- slim_script(
  slim_block_init_minimal(),
  slim_block(1, 100, {
    sim.outputFull()
  })
)
script
end_gen(script)
end_gen(script)[2] <- 1000
script
```

`evaluate` *SLiM method evaluate*

Description

Documentation for SLiM function `evaluate`, which is a method of the SLiM class `InteractionType`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
evaluate(subpops)
```

Arguments

`subpops`	An object of type integer or Subpopulation object. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 693.

Snapshots model state in preparation for the use of the interaction, for the receiver and exerter subpopulations specified by subpops. The subpopulations may be supplied either as integer IDs, or as Subpopulation objects. This method will discard all previously cached data for the subpopulation(s), and will cache the current spatial positions of all individuals they contain (so that the spatial positions of those individuals may then change without disturbing the state of the interaction at the moment of evaluation). It will also cache which individuals in the subpopulation are eligible to act as exerters, according to the configured exerter constraints, but it will not cache such eligibility information for receiver constraints (which are applied at the time a spatial query is made). Particular interaction distances and strengths are not computed by evaluate(), and interaction() callbacks will not be called in response to this method; that work is deferred until required to satisfy a query (at which point the tick and cycle counters may have advanced, so be careful with the tick ranges used in defining interaction() callbacks). You must explicitly call evaluate() at an appropriate time in the tick cycle before the interaction is used, but after any relevant changes have been made to the population. SLiM will invalidate any existing interactions after any portion of the tick cycle in which new individuals have been born or existing individuals have died. In a WF model, this occurs just before late() events execute (see the WF tick cycle diagram in chapter 23), so late() events are often the appropriate place to put evaluate() calls, but first() or early() events can work too if the interaction is not needed until that point in the tick cycle anyway. In nonWF models, on the other hand, new offspring are produced just before early() events and then individuals die just before late() events (see the nonWF tick cycle diagram in chapter 24), so interactions will be invalidated twice during each tick cycle. This means that in a nonWF model, an interaction that influences reproduction should usually be evaluated in a first() event, while an interaction that influences fitness or mortality should usually be evaluated in an early() event (and an interaction that affects both may need to be evaluated at both times). If an interaction is never evaluated for a given subpopulation, it is guaranteed that there will be essentially no memory or computational overhead associated with the interaction for that subpopulation. Furthermore, attempting to query an interaction for a receiver or exerter in a subpopulation that has not been evaluated is guaranteed to raise an error.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other InteractionType: IT, clippedIntegral(), distanceFromPoint(), distance(), drawByStrength(), interactingNeighborCount(), interactionDistance(), localPopulationDensity(), nearestInteractingNei nearestNeighborsOfPoint(), nearestNeighbors(), neighborCountOfPoint(), neighborCount(), setConstraints(), setInteractionFunction(), strength(), testConstraints(), totalOfNeighborStrengths unevaluate()

`exp` *SLiM method exp*

Description

Documentation for SLiM function `exp`, which is a method of the SLiM class SpatialMap. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
exp(void)
```

Arguments

`void`	An object of type . See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 714.

Exponentiates the values of the spatial map. More precisely, each grid value x of the target spatial map is exponentiated - replaced by the value ex. The target spatial map is returned, to allow easy chaining of operations.

Value

An object of type SpatialMap object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other SpatialMap: SM, add(), blend(), changeColors(), changeValues(), divide(), gridValues(), interpolate(), mapColor(), mapImage(), mapValue(), multiply(), power(), range(), rescale(), sampleImprovedNearbyPoint(), sampleNearbyPoint(), smooth(), subtract()

---

first *SLiM first() callback*

Description

This callback specifies the code should be called first (before anything else) in the simulation cycle. For details on exactly when first(), early() and early() callbacks are run during a simulation see SLiM Manual: page 541 for "WF" models, or SLiM Manual: page 549 for "nonWF" models.

Usage

```
first()
```

Value

None

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other callbacks: early(), fitnessEffect(), fitness(), initialize(), interaction(), late(), mateChoice(), modifyChild(), mutationEffect(), mutation(), recombination(), reproduction(), slim_callbacks(), survival()

Examples

```
slim_block(1, early(), {
  sim.addSubpop("p1", 100)
})
```

`fitness` *SLiM fitness() callback*

Description

This callback specifies that a code block is providing logic to decide the fitness of a mutation in an individual. The callback should return a the relative fitness value of the mutation, where a value of 1.0 is neutral. This is called once per mutation per individual per generation in which it is active. If `mut_id_type = NULL` then the callback will be called only once per individual and should return a global relative fitness value that is independent of mutations the individual possesses. see SLiM Manual: page 596

Usage

```
fitness(mut_type_id, subpop_id)
```

Arguments

`mut_type_id`	The id of the mutationType to which this callback should apply. Can be an integer 1, 2, etc., or character ”m1”, ”m2”, etc.
`subpop_id`	The id(s) of the subpopulation(s) to which this callback should apply. Can be an integer 1, 2, etc., or character ”p1”, ”p2”, etc.

Details

Global variables available in reproduction callbacks:

mut A Mutation object, the mutation whose relative fitness is being evaluated

homozygous A value of T (the mutation is homozygous), F (heterozygous), or NULL (it is paired with a null chromosome, which can occur with sex chromosomes)

relFitness The default relative fitness value calculated by SLiM

individual The individual carrying this mutation (an object of class Individual)

genome1 One genome of the individual carrying this mutation

genome2 The other genome of that individual

subpop The subpopulation in which that individual lives

Value

None

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other callbacks: `early()`, `first()`, `fitnessEffect()`, `initialize()`, `interaction()`, `late()`, `mateChoice()`, `modifyChild()`, `mutationEffect()`, `mutation()`, `recombination()`, `reproduction()`, `slim_callbacks()`, `survival()`

Examples

```
slim_block(fitness(m2), {
  ## sets up frequency dependent selection for m2
  return(1.5 - sim.mutationFrequencies(p1, mut))
})
```

`fitnessEffect` *SLiM fitnessEffect() callback*

Description

A fitnessEffect() callback is called by SLiM when it is determining the fitness of an individual – typically, but not always, once per tick during the fitness calculation tick cycle stage. Normally, the fitness of a given individual is determined by multiplying together the fitness effects of all mutations possessed by that individual (see section 25.2 for further discussion). Supplying a fitnessEffect() callback allows you to add another multiplicative fitness effect into that calculation. As with mutationEffect() callbacks, the value returned by fitnessEffect() callbacks is a fitness effect, so 1.0 is neutral. For details see SLiM Manual: page 715

Usage

```
fitnessEffect(subpop_id)
```

Arguments

`subpop_id`	The id(s) of the subpopulation(s) to which this callback should apply. Can be an integer 1, 2, etc., or character "p1", "p2", etc.

Details

Global variables available in reproduction callbacks:

individual The individual carrying this mutation (an object of class Individual)

subpop The subpopulation in which that individual lives

Value

None

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other callbacks: `early()`, `first()`, `fitness()`, `initialize()`, `interaction()`, `late()`, `mateChoice()`, `modifyChild()`, `mutationEffect()`, `mutation()`, `recombination()`, `reproduction()`, `slim_callbacks()`, `survival()`

Examples

```
slim_block(fitnessEffect(p3), {
  # multiplies all individual's fitness by 0.75
  return(0.75)
})
```

`flush` *SLiM method flush*

Description

Documentation for SLiM function `flush`, which is a method of the SLiM class `LogFile`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
flush(void)
```

Arguments

`void`	An object of type . See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 702.

Flushes all buffered data to the output file, synchronously. This will make the contents of the file on disk be up-to-date with the running simulation. Flushing frequently may entail a small performance penalty. More importantly, if .gz compression has been requested with compress=T the size of the resulting file will be larger - potentially much larger - if flush() is called frequently. Note that automatic periodic flushing can be requested with the flushInterval parameter to createLogFile().

Value

An object of type void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or logical.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other LogFile: LF, addCustomColumn(), addCycleStage(), addCycle(), addKeysAndValuesFrom(), addMeanSDColumns(), addPopulationSexRatio(), addPopulationSize(), addSubpopulationSexRatio(), addSubpopulationSize(), addSuppliedColumn(), addTick(), clearKeysAndValues(), logRow(), setFilePath(), setLogInterval(), setSuppliedValue(), setValue(), willAutolog()

G *Genome*

Description

Documentation for Genome class from SLiM

Details

This class represents one full genome of an individual (one of the two genomes contained by a diploid individual, that is, in the way that SLiM uses the term), composed of the mutations carried by that individual. Section 1.5.1 presents an overview of the conceptual role of this class. This class has the following methods (functions):

- addMutations

- addNewDrawnMutation
- addNewMutation
- containsMarkerMutation
- containsMutations
- countOfMutationsOfType
- mutationCountsInGenomes
- mutationFrequenciesInGenomes
- mutationsOfType
- nucleotides
- output
- outputMS
- outputVCF
- positionsOfMutationsOfType
- readFromMS
- readFromVCF
- removeMutations
- sumOfMutationsOfType

This class has the following properties:

genomePedigreeID A property of type integer. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** If pedigree tracking is turned on with initializeSLiMOptions(keepPedigrees=T), genomePedigreeID is a unique non-negative identifier for each genome in a simulation, never reused throughout the duration of the simulation run. Furthermore, the genomePedigreeID of a given genome will be equal to either (2*pedigreeID) or (2*pedigreeID + 1) of the individual that the genome belongs to (the former for the first genome of the individual, the latter for the second genome of the individual); this invariant relationship is guaranteed. If pedigree tracking is not enabled, this property is unavailable.

genomeType A property of type string. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The type of chromosome represented by this genome; one of "A", "X", or "Y".

individual A property of type Individual object. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The Individual object to which this genome belongs.

isNullGenome A property of type logical. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** T if the genome is a "null" genome, F if it is an ordinary genome object. When a sex chromosome (X or Y) is simulated, the other sex chromosome also exists in the simulation, but it is a "null" genome that does not carry any mutations. Instead, it is a placeholder, present to allow SLiM's code to operate in much the same way as it does when an autosome is simulated. Null genomes should not be accessed or manipulated.

mutations A property of type Mutation object. This property is a constant, so it is not modifiable. **Property Description:** All of the Mutation objects present in this genome.

tag A property of type integer. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** A user-defined integer value. The value of tag is initially undefined, and it is an error to try to read it; if you wish it to have a defined value, you must arrange that yourself by explicitly setting its value prior to using it elsewhere in your code. The value of tag is not used by SLiM; it is free for you to use. Note that the Genome objects used by SLiM are new with every new individual, so the tag value of each new offspring generated in each tick will be initially undefined.

See Also

Other Genome: `addMutations()`, `addNewDrawnMutation()`, `addNewMutation()`, `containsMarkerMutation()`, `containsMutations()`, `countOfMutationsOfType()`, `mutationCountsInGenomes()`, `mutationFrequenciesInGe`
`mutationsOfType()`, `nucleotides()`, `outputMS()`, `outputVCF()`, `output()`, `positionsOfMutationsOfType()`, `readFromMS()`, `readFromVCF()`, `removeMutations()`, `sumOfMutationsOfType()`

GE *GenomicElement*

Description

Documentation for GenomicElement class from SLiM

Details

This class represents a genomic element of a particular genomic element type, with a start and end; the chromosome is composed of a series of such genomic elements. Section 1.5.4 presents an overview of the conceptual role of this class. This class has the following methods (functions):

- `setGenomicElementType`

This class has the following properties:

endPosition A property of type integer. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The last position in the chromosome contained by this genomic element.

genomicElementType A property of type GenomicElementType object. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The GenomicElementType object that defines the behavior of this genomic element.

startPosition A property of type integer. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The first position in the chromosome contained by this genomic element.

tag A property of type integer. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** A user-defined integer value. The value of tag is initially undefined, and it is an error to try to read it; if you wish it to have a defined value, you must arrange that yourself by explicitly setting its value prior to using it elsewhere in your code. The value of tag is not used by SLiM; it is free for you to use.

See Also

Other GenomicElement: `setGenomicElementType()`

`genomicElementTypesWithIDs`
SLiM method genomicElementTypesWithIDs

Description

Documentation for SLiM function `genomicElementTypesWithIDs`, which is a method of the SLiM class `Community`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
genomicElementTypesWithIDs(ids)
```

Arguments

`ids`	An object of type integer. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 666.

Find and return the GenomicElementType objects with id values matching the values in ids. If no matching GenomicElementType object can be found with a given id, an error results.

Value

An object of type GenomicElementType object.

Copyright

Author(s)

Benjamin C Haller (`<bhaller@benhaller.com>`) and Philipp W Messer (`<messer@cornell.edu>`)

See Also

Other Community: Co, createLogFile(), deregisterScriptBlock(), interactionTypesWithIDs(), mutationTypesWithIDs(), outputUsage(), registerEarlyEvent(), registerFirstEvent(), registerInteractionCallback(), registerLateEvent(), rescheduleScriptBlock(), scriptBlocksWithIDs() simulationFinished(), speciesWithIDs(), subpopulationsWithIDs(), usage()

GET *GenomicElementType*

Description

Documentation for GenomicElementType class from SLiM

Details

This class represents a type of genomic element, with particular mutation types. The genomic element types currently defined in the simulation are defined as global constants with the same names used in the SLiM input file - g1, g2, and so forth. Section 1.5.4 presents an overview of the conceptual role of this class. This class has the following methods (functions):

- setMutationFractions
- setMutationMatrix

This class has the following properties:

color A property of type string. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** The color used to display genomic elements of this type in SLiMgui. Outside of SLiMgui, this property still exists, but is not used by SLiM. Colors may be specified by name, or with hexadecimal RGB values of the form "#RRGGBB" (see the Eidos manual). If color is the empty string, "", SLiMgui's default color scheme is used; this is the default for new GenomicElementType objects.

id A property of type integer. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The identifier for this genomic element type; for genomic element type g3, for example, this is 3.

mutationFractions A property of type float. This property is a constant, so it is not modifiable. **Property Description:** For each MutationType represented in this genomic element type, this property has the corresponding fraction of all mutations that will be drawn from that MutationType.

mutationMatrix A property of type float. This property is a constant, so it is not modifiable. **Property Description:** The nucleotide mutation matrix used for this genomic element type, set up by initializeGenomicElementType() and setMutationMatrix(). This property is only defined in nucleotide-based models; it is unavailable otherwise.

mutationTypes A property of type MutationType object. This property is a constant, so it is not modifiable. **Property Description:** The MutationType instances used by this genomic element type.

species A property of type Species object. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The species to which the target object belongs.

tag A property of type integer. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** A user-defined integer value. The value of tag is initially undefined, and it is an error to try to read it; if you wish it to have a defined value, you must arrange that yourself by explicitly setting its value prior to using it elsewhere in your code. The value of tag is not used by SLiM; it is free for you to use. See also the getValue() and setValue() methods (provided by the Dictionary class; see the Eidos manual), for another way of attaching state to genomic element types.

See Also

Other GenomicElementType: `setMutationFractions()`, `setMutationMatrix()`

`get_block` *Extract a single code block from a* `slimr_script`

Description

Extract a single code block from a `slimr_script`

Usage

```
get_block(x, i)
```

Arguments

`x`	`slimr_script` object to extract block from
`i`	The block to extract. Can be either an integer specifying what block(s) index to extract, or a character, in which case it pulls the block(s) with the corresponding name.

Examples

```
script <- slim_script(
  slim_block_init_minimal(),
  slim_block_finish(100)
)
get_block(script, "block_init")
get_block(script, 2)
```

`get_slim_call` *Get the call information for running SLiM. Doubles as a check for SLiM availability.*

Description

Function to test if SLiM is available on user's system and return the correct system call to execute it.

Usage

```
get_slim_call()
```

`gridValues` *SLiM method gridValues*

Description

Documentation for SLiM function `gridValues`, which is a method of the SLiM class `SpatialMap`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
gridValues(void)
```

Arguments

`void`	An object of type . See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 714.

Returns the values for the spatial map's grid as a vector (for a 1D map), a matrix (for a 2D map), or an array (for a 3D map). The form and orientation of the returned values is such that it could be used to create a new spatial map, with defineSpatialMap(), which would be identical to the original.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other SpatialMap: SM, add(), blend(), changeColors(), changeValues(), divide(), exp(), interpolate(), mapColor(), mapImage(), mapValue(), multiply(), power(), range(), rescale(), sampleImprovedNearbyPoint(), sampleNearbyPoint(), smooth(), subtract()

In *Individual*

Description

Documentation for Individual class from SLiM

Details

This class represents a single simulated individual. Individuals in SLiM are diploid, and thus contain two Genome objects. Most functionality in SLiM is contained in the Genome class; the Individual class is mostly a convenient way to treat the pairs of genomes associated with an individual as a single object, and to associate a tag value with individuals. Section 1.5.1 presents an overview of the conceptual role of this class. This class has the following methods (functions):

- containsMutations
- countOfMutationsOfType
- relatedness
- setSpatialPosition
- sharedParentCount
- sumOfMutationsOfType
- uniqueMutationsOfType

This class has the following properties:

age A property of type integer. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** The age of the individual, measured in cycles. A newly generated offspring individual will have an age of 0 in the same tick in which it was created. The age of every individual is incremented by one at the same point that its species cycle counter is incremented, at the end of the tick cycle, if and only if its species was active in that tick. The age of individuals may be changed; usually this only makes sense when setting up the initial state of a model, however.

color A property of type string. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** The color used to display the individual in SLiMgui. Outside of SLiMgui, this property still exists, but is not used by SLiM. Colors may be specified by name, or with hexadecimal RGB values of the form "#RRGGBB" (see the Eidos manual). If color is the empty string, "", SLiMgui's default (fitness-based) color scheme is used; this is the default for new Individual objects. Note that named colors will be converted to RGB internally, so the value of this property will always be a hexadecimal RGB color string (or "").

fitnessScaling A property of type float. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** A float scaling factor applied to the individual's fitness (i.e., the fitness value computed for the individual will be multiplied by this value). This provides a simple, fast way to modify the fitness of an individual; conceptually it is similar to returning a fitness effect for the individual from a fitnessEffect() callback, but without the complexity and performance overhead of implementing such a callback. To scale the fitness of all individuals in a subpopulation by the same factor, see the fitnessScaling property of Subpopulation. The value of fitnessScaling is reset to 1.0 every tick, so that any scaling factor set lasts for only a single tick. This reset occurs immediately after fitness values are calculated, in both WF and nonWF models.

genomes A property of type Genome object. This property is a constant, so it is not modifiable. **Property Description:** The pair of Genome objects associated with this individual. If only one of the two genomes is desired, the genome1 or genome2 property may be used.

genomesNonNull A property of type Genome object. This property is a constant, so it is not modifiable. **Property Description:** The pair of Genome objects associated with this individual, as with the genomes property, if both are not null genomes. If one or both are null genomes, the null genomes are excluded from the returned vector. This is a convenience shorthand, sometimes useful in models that involve null genomes.

genome1 A property of type Genome object. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The first Genome object associated with this individual. This property is particularly useful when you want the first genome from each of a vector of individuals, as often arises in haploid models.

genome2 A property of type Genome object. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The second Genome object associated with this individual. This property is particularly useful when you want the second genome from each of a vector of individuals, as often arises in haploid models.

index A property of type integer. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The index of the individual

in the individuals vector of its Subpopulation.

meanParentAge A property of type float. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The average age of the parents of this individual, measured in cycles. Parentless individuals will have a meanParentAge of 0.0. The mean parent age is determined when a new offspring is generated, from the age property of the parent or parents involved in generating the offspring. For addRecombinant() that is somewhat complex; see that method for details.

migrant A property of type logical. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** Set to T if the individual is a recent migrant, F otherwise. The definition of "recent" depends upon the model type (WF or nonWF). In WF models, this flag is set at the point when a new child is generated if it is a migrant (i.e., if its source subpopulation is not the same as its subpopulation), and remains valid, with the same value, for the rest of the individual's lifetime. In nonWF models, this flag is F for all new individuals, is set to F in all individuals at the end of the reproduction tick cycle stage, and is set to T on all individuals moved to a new subpopulation by takeMigrants() or a survival() callback; the T value set by takeMigrants() or survival() will remain until it is reset at the end of the next reproduction tick cycle stage.

pedigreeID A property of type integer. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** If pedigree tracking is turned on with initializeSLiMOptions(keepPedigrees=T), pedigreeID is a unique non-negative identifier for each individual in a simulation, never re-used throughout the duration of the simulation run. If pedigree tracking is not enabled, this property is unavailable.

pedigreeParentIDs A property of type integer. This property is a constant, so it is not modifiable. **Property Description:** If pedigree tracking is turned on with initializeSLiMOptions(keepPedigrees=T), pedigreeParentIDs contains the values of pedigreeID that were possessed by the parents of an individual; it is thus a vector of two values. If pedigree tracking is not enabled, this property is unavailable. Parental values may be -1 if insufficient ticks have elapsed for that information to be available (because the simulation just started, or because a subpopulation is new).

pedigreeGrandparentIDs A property of type integer. This property is a constant, so it is not modifiable. **Property Description:** If pedigree tracking is turned on with initializeSLiMOptions(keepPedigrees=T), pedigreeGrandparentIDs contains the values of pedigreeID that were possessed by the grandparents of an individual; it is thus a vector of four values. If pedigree tracking is not enabled, this property is unavailable. Grandparental values may be -1 if insufficient ticks have elapsed for that information to be available (because the simulation just started, or because a subpopulation is new).

reproductiveOutput A property of type integer. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** If pedigree tracking is turned on with initializeSLiMOptions(keepPedigrees=T), reproductiveOutput contains the number of offspring for which this individual has been a parent. If pedigree tracking is not enabled, this property is unavailable. If an individual is a parent by cloning or selfing, or as both parents for a biparental mating, this value is incremented by two. Involvement of an individual as a parent for an addRecombinant()

call does not change this property's value, since the reproductive contribution in that case is unclear; one must conduct separate bookkeeping for that case if necessary. See also the Subpopulation property lifetimeReproductiveOutput.

sex A property of type string. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The sex of the individual. This will be "H" if sex is not enabled in the simulation (i.e., for hermaphrodites), otherwise "F" or "M" as appropriate.

spatialPosition A property of type float. This property is a constant, so it is not modifiable. **Property Description:** The spatial position of the individual. The length of the spatialPosition property (the number of coordinates in the spatial position of an individual) depends upon the spatial dimensionality declared with initializeSLiMOptions(). If the spatial dimensionality is zero (as it is by default), it is an error to access this property. The elements of this property are identical to the values of the x, y, and z properties (if those properties are encompassed by the spatial dimensionality of the simulation). In other words, if the declared dimensionality is "xy", the individual.spatialPosition property is equivalent to c(individual.x, individual.y); individual.z is not used since it is not encompassed by the simulation's dimensionality. This property cannot be set, but the setSpatialPosition() method may be used to achieve the same thing.

subpopulation A property of type Subpopulation object. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The Subpopulation object to which the individual belongs.

tag A property of type integer. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** A user-defined integer value (as opposed to tagF, which is of type float). The value of tag is initially undefined, and it is an error to try to read it; if you wish it to have a defined value, you must arrange that yourself by explicitly setting its value prior to using it elsewhere in your code. The value of tag is not used by SLiM; it is free for you to use. See also the getValue() and setValue() methods (provided by the Dictionary class; see the Eidos manual), for another way of attaching state to individuals. Note that the Individual objects used by SLiM are new for every new offspring, so the tag value of each new offspring generated in each tick will be initially undefined.

tagF A property of type float. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** A user-defined float value (as opposed to tag, which is of type integer). The value of tagF is initially undefined, and it is an error to try to read it; if you wish it to have a defined value, you must arrange that yourself by explicitly setting its value prior to using it elsewhere in your code. The value of tagF is not used by SLiM; it is free for you to use. See also the getValue() and setValue() methods (provided by the Dictionary class; see the Eidos manual), for another way of attaching state to individuals. Note that at present, although many classes in SLiM have an integer-type tag property, only Individual has a float-type tagF property, because attaching model state to individuals seems to be particularly common and useful. If a tagF property would be helpful on another class, it would be easy to add. See the description of the tag property above for additional comments.

tagL0 A property of type logical. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** A user-defined logical value (see also tag and tagF). The value of tagL0 is initially undefined, and it is an error to try

to read it; if you wish it to have a defined value, you must arrange that yourself by explicitly setting its value prior to using it elsewhere in your code. The value of tagL0 is not used by SLiM; it is free for you to use. See also the getValue() and setValue() methods (provided by the Dictionary class; see the Eidos manual), for another way of attaching state to individuals.

tagL1 A property of type logical. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** A user-defined logical value (see also tag and tagF). The value of tagL1 is initially undefined, and it is an error to try to read it; if you wish it to have a defined value, you must arrange that yourself by explicitly setting its value prior to using it elsewhere in your code. The value of tagL1 is not used by SLiM; it is free for you to use. See also the getValue() and setValue() methods (provided by the Dictionary class; see the Eidos manual), for another way of attaching state to individuals.

tagL2 A property of type logical. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** A user-defined logical value (see also tag and tagF). The value of tagL2 is initially undefined, and it is an error to try to read it; if you wish it to have a defined value, you must arrange that yourself by explicitly setting its value prior to using it elsewhere in your code. The value of tagL2 is not used by SLiM; it is free for you to use. See also the getValue() and setValue() methods (provided by the Dictionary class; see the Eidos manual), for another way of attaching state to individuals.

tagL3 A property of type logical. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** A user-defined logical value (see also tag and tagF). The value of tagL3 is initially undefined, and it is an error to try to read it; if you wish it to have a defined value, you must arrange that yourself by explicitly setting its value prior to using it elsewhere in your code. The value of tagL3 is not used by SLiM; it is free for you to use. See also the getValue() and setValue() methods (provided by the Dictionary class; see the Eidos manual), for another way of attaching state to individuals.

tagL4 A property of type logical. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** A user-defined logical value (see also tag and tagF). The value of tagL4 is initially undefined, and it is an error to try to read it; if you wish it to have a defined value, you must arrange that yourself by explicitly setting its value prior to using it elsewhere in your code. The value of tagL4 is not used by SLiM; it is free for you to use. See also the getValue() and setValue() methods (provided by the Dictionary class; see the Eidos manual), for another way of attaching state to individuals.

uniqueMutations A property of type Mutation object. This property is a constant, so it is not modifiable. **Property Description:** All of the Mutation objects present in this individual. Mutations present in both genomes will occur only once in this property, and the mutations will be given in sorted order by position, so this property is similar to sortBy(unique(individual.genomes.mutations), "position"). It is not identical to that call, only because if multiple mutations exist at the exact same position, they may be sorted differently by this method than they would be by sortBy(). This method is provided primarily for speed; it executes much faster than the Eidos equivalent above. Indeed, it is faster than just individual.genomes.mutations, and gives uniquing and sorting on top of that, so it is advantageous unless duplicate entries for homozygous mutations are actually needed.

x A property of type float. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** A user-defined float value. The value of x is initially undefined (i.e., has an effectively random value that could be different every time you run your model); if you wish it to have a defined value, you must arrange that yourself by explicitly setting its value prior to using it elsewhere in your code. The value of x is not used by SLiM unless the optional "continuous space" facility is enabled with the dimensionality parameter to initializeSLiMOptions(), in which case x will be understood to represent the x coordinate of the individual in space. If continuous space is not enabled, you may use x as an additional tag value of type float.

xy A property of type float. This property is a constant, so it is not modifiable. **Property Description:** This property provides joint read-only access to the x and y properties; they are returned as a twoelement float vector. This can be useful in complex spatial models in which the spatiality of interactions/maps differs from the overall dimensionality of the model. See the documentation for the separate properties x and y for further comments.

xyz A property of type float. This property is a constant, so it is not modifiable. **Property Description:** This property provides joint read-only access to the x, y, and z properties; they are returned as a threeelement float vector. This can be useful in complex spatial models in which the spatiality of interactions/maps differs from the overall dimensionality of the model. See the documentation for the separate properties x, y, and z for further comments.

xz A property of type float. This property is a constant, so it is not modifiable. **Property Description:** This property provides joint read-only access to the x and z properties; they are returned as a twoelement float vector. This can be useful in complex spatial models in which the spatiality of interactions/maps differs from the overall dimensionality of the model. See the documentation for the separate properties x and z for further comments.

y A property of type float. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** A user-defined float value. The value of y is initially undefined (i.e., has an effectively random value that could be different every time you run your model); if you wish it to have a defined value, you must arrange that yourself by explicitly setting its value prior to using it elsewhere in your code. The value of y is not used by SLiM unless the optional "continuous space" facility is enabled with the dimensionality parameter to initializeSLiMOptions(), in which case y will be understood to represent the y coordinate of the individual in space (if the dimensionality is "xy" or "xyz"). If continuous space is not enabled, or the dimensionality is not "xy" or "xyz", you may use y as an additional tag value of type float.

yz A property of type float. This property is a constant, so it is not modifiable. **Property Description:** This property provides joint read-only access to the y and z properties; they are returned as a twoelement float vector. This can be useful in complex spatial models in which the spatiality of interactions/maps differs from the overall dimensionality of the model. See the documentation for the separate properties y and z for further comments.

z A property of type float. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** A user-defined float value. The value of z is initially undefined (i.e., has an effectively random value that could be different every

time you run your model); if you wish it to have a defined value, you must arrange that yourself by explicitly setting its value prior to using it elsewhere in your code. The value of z is not used by SLiM unless the optional "continuous space" facility is enabled with the dimensionality parameter to initializeSLiMOptions(), in which case z will be understood to represent the z coordinate of the individual in space (if the dimensionality is "xyz"). If continuous space is not enabled, or the dimensionality is not "xyz", you may use z as an additional tag value of type float.

See Also

Other Individual: `containsMutations()`, `countOfMutationsOfType()`, `relatedness()`, `setSpatialPosition()`, `sharedParentCount()`, `sumOfMutationsOfType()`, `uniqueMutationsOfType()`

`individualsWithPedigreeIDs`

SLiM method individualsWithPedigreeIDs

Description

Documentation for SLiM function `individualsWithPedigreeIDs`, which is a method of the SLiM class `Species`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
individualsWithPedigreeIDs(pedigreeIDs, subpops)
```

Arguments

`pedigreeIDs`	An object of type integer. See details for description.
`subpops`	An object of type null or integer or Subpopulation object. The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 720.

Looks up individuals by pedigree ID, optionally within specific subpopulations. Pedigree tracking must be turned on with initializeSLiMOptions(keepPedigrees=T) to use this method, otherwise an error will result. This method is vectorized; more than one pedigree id may be passed in pedigreeID, in which case the returned vector will contain all of the individuals for which a match was found (in the same order in which they were supplied). If a given id is not found, the returned vector will contain no entry for that id (so the length of the returned vector may not match the length of pedigreeIDs). If none of the given ids were found, the returned vector will be object<Individual>(0), an empty object vector of class Individual. If you have more than one pedigree ID to look up, calling this method

just once, in vectorized fashion, may be much faster than calling it once for each ID, due to internal optimizations. To find individuals within all subpopulations, pass the default of NULL for subpops. If you are interested only in matches within a specific subpopulation, pass that subpopulation for subpops; that will make the search faster. Similarly, if you know that a particular subpopulation is the most likely to contain matches, you should supply that subpopulation first in the subpops vector so that it will be searched first; the supplied subpopulations are searched in order. Subpopulations may be supplied either as integer IDs, or as Subpopulation objects.

Value

An object of type Individual object.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Species: Sp, addSubpopSplit(), addSubpop(), countOfMutationsOfType(), killIndividuals(), mutationCounts(), mutationFrequencies(), mutationsOfType(), outputFixedMutations(), outputFull(), outputMutations(), readFromPopulationFile(), recalculateFitness(), registerFitnessEffectCallback(), registerMateChoiceCallback(), registerModifyChildCallback(), registerMutationCallback(), registerMutationEffectCallback(), registerRecombinationCallback(), registerReproductionCallback(), registerSurvivalCallback(), simulationFinished(), skipTick(), subsetMutations(), treeSeqCoalesced(), treeSeqOutput(), treeSeqRememberIndividuals(), treeSeqSimplify()

Init *Initialize*

Description

Documentation for Initialize class from SLiM

Details

Before a SLiM simulation can be run, the various classes underlying the simulation need to be set up with an initial configuration. Simulation configuration in SLiM is done in initialize() callbacks that run prior to the beginning of simulation execution. Eidos callbacks are discussed more broadly in chapter 26, but for our present purposes, the idea is very simple. In your input file, you can write something like this: initialize()

...

The initialize() declaration specifies that the script block is to be executed as an initialize() callback before the simulation starts. The script between the braces would set up various aspects of the simulation by calling initialization functions. These are SLiM functions that may be called only in an initialize() callback, and their names begin with initialize to mark them clearly as such. You may also use other Eidos functionality in these callbacks; for example, you might automate generating a complex genetic structure containing many genes by using a for loop. In general, it is required for a species to set up its genetic structure in an initialize() callback with calls to initializeMutationRate(), initializeRecombinationRate(), initializeMutationType(), initializeGenomicElementType(), and initializeGenomicElement(); species must call all of these, setting up at least one mutation type, at least one genomic element type, and at least one genomic element. The exception to this general rule is for species that have no genetics at all - species that are modeled purely on an ecological/behavioral level. Such species may be defined by calling none of those initialization functions; in this case, SLiM will default to a zero-length chromosome with mutation and recombination rates of zero. A middle ground between these two configuration paths is not allowed; either a species has no genetics, or it fully defines its genetics. One thing worth mentioning is that in the context of an initialize() callback, the sim global representing the species being simulated is not defined. This is because the state of the simulation is not yet constructed fully, and accessing partially constructed state would not be safe. (Similarly, in multispecies models, the community object and the objects representing individual species are not yet defined.) The above initialize() callback syntax implicitly declares a single species, with the default name of sim, and therefore sets up a single-species model. It is also possible to explicitly declare a species, which is done with this extended syntax (using a species name of fox as an example): species fox initialize() ... This sets up a multispecies model (although it might, in fact, declare only a single species, fox; the term "multispecies", in SLiM parlance, really means "explicitly declared species", but multispecies models almost always do contain multiple species, so the distinction is unimportant). See section 1.9 and chapter 19 for further discussion of multispecies models; in most respects they work identically to single-species models, so we will tend to focus on the single-species case in the reference documentation, with a species name of sim, for simplicity and clarity. In single-species models all initialization can be done in a single initialize() callback (or you can have more than one, if you wish). In multispecies models, each species must be initialized with its own callback(s), as shown above. In addition, multispecies models also support an optional community-level initialization callback that is declared as follows: species all initialize() ... These callbacks, technically called non-species-specific initialize() callbacks, provide a place for community-level initialization to occur. They are run before any species-specific initialize() callbacks are run, so you might wish to set up all of your model parameters in one, providing a single location for all parameters. In multispecies models, the initialization functions initializeSLiMModelType() and initializeInteractionType() may only be called from a nonspecies- specific initialize()

callback, since those aspects of model configuration span the entire community. In single-species models, these functions may be called from an ordinary initialize() callback for simplicity and backward compatibility. This class has the following methods (functions):

- initializeAncestralNucleotides
- initializeGeneConversion
- initializeGenomicElement
- initializeGenomicElementType
- initializeHotspotMap
- initializeInteractionType
- initializeMutationRate
- initializeMutationType
- initializeMutationTypeNuc
- initializeRecombinationRate
- initializeSex
- initializeSLiMModelType
- initializeSLiMOptions
- initializeSpecies
- initializeTreeSeq

This class has the following properties:

None. This class has no properties.

See Also

Other Initialize: initializeAncestralNucleotides(), initializeGeneConversion(), initializeGenomicElementType(), initializeGenomicElement(), initializeHotspotMap(), initializeInteractionType(), initializeMutationRate(), initializeMutationTypeNuc(), initializeMutationType(), initializeRecombinationRate(), initializeSLiMModelType(), initializeSLiMOptions(), initializeSex(), initializeSpecies(), initializeTreeSeq()

---

initialize *SLiM initialize() callback*

Description

This callback specifies the code should be called to set up the simulation. For details on what you can do in an initialize() callback code blocks see SLiM Manual: page 516

Usage

```
initialize()
```

Value

None

Copyright

This is documentation for a function in the SLiM software, and has been modified from the official manual, which can be found here: http://benhaller.com/slim/SLiM_Manual.pdf. This documentation is Copyright © 2016-2020 Philipp Messer. All rights reserved. More information about SLiM can be found on the official website: https://messerlab.org/slim/

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other callbacks: `early()`, `first()`, `fitnessEffect()`, `fitness()`, `interaction()`, `late()`, `mateChoice()`, `modifyChild()`, `mutationEffect()`, `mutation()`, `recombination()`, `reproduction()`, `slim_callbacks()`, `survival()`

Examples

```
slim_block(initialize(), {
  initializeMutationRate(1e-7)
  initializeMutationType("m1", 0.5, "f", 0.0)
  initializeGenomicElementType("g1", m1, 1.0)
  initializeGenomicElement(g1, 0, 99999)
  initializeRecombinationRate(1e-8)
})
```

`initializeAncestralNucleotides`

SLiM method initializeAncestralNucleotides

Description

Documentation for SLiM function `initializeAncestralNucleotides`, which is a method of the SLiM class `Initialize`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
initializeAncestralNucleotides(sequence)
```

Arguments

sequence	An object of type integer or string. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 647.

This function, which may be called only in nucleotide-based models (see section 1.8), supplies an ancestral nucleotide sequence for the model. The sequence parameter may be an integer vector providing nucleotide values (A=0, C=1, G=2, T=3), or a string vector providing single-character nucleotides ("A", "C", "G", "T"), or a singleton string providing the sequence as one string ("ACGT..."), or a singleton string providing the filesystem path of a FASTA file which will be read in to provide the sequence (if the file contains than one sequence, the first sequence will be used). Only A/C/G/T nucleotide values may be provided; other symbols, such as those for amino acids, gaps, or nucleotides of uncertain identity, are not allowed. The two semantic meanings of sequence that involve a singleton string value are distinguished heuristically; a singleton string that contains only the letters ACGT will be assumed to be a nucleotide sequence rather than a filename. The length of the ancestral sequence is returned. A utility function, randomNucleotides(), is provided by SLiM to assist in generating simple random nucleotide sequences; see section 25.18.1.

Value

An object of type integer. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Initialize: Init, initializeGeneConversion(), initializeGenomicElementType(), initializeGenomicElement(), initializeHotspotMap(), initializeInteractionType(), initializeMutationRate(), initializeMutationTypeNuc(), initializeMutationType(), initializeRecombinationRate(), initializeSLiMModelType(), initializeSLiMOptions(), initializeSex(), initializeSpecies(), initializeTreeSeq()

Examples

```
## This just brings up the documentation:
initializeAncestralNucleotides()
```

`initializeGeneConversion`
SLiM method initializeGeneConversion

Description

Documentation for SLiM function `initializeGeneConversion`, which is a method of the SLiM class `Initialize`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
initializeGeneConversion(
  nonCrossoverFraction,
  meanLength,
  simpleConversionFraction,
  bias,
  redrawLengthsOnFailure
)
```

Arguments

`nonCrossoverFraction`
: An object of type numeric or numeric or numeric or numeric or logical. Must be of length 1 (a singleton). See details for description.

`meanLength`
: An object of type numeric or numeric or numeric or numeric or logical. Must be of length 1 (a singleton). See details for description.

`simpleConversionFraction`
: An object of type numeric or numeric or numeric or numeric or logical. Must be of length 1 (a singleton). See details for description.

`bias`
: An object of type numeric or numeric or numeric or numeric or logical. Must be of length 1 (a singleton). The default value is `0`. See details for description.

`redrawLengthsOnFailure`
: An object of type numeric or numeric or numeric or numeric or logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 647.

Calling this function switches the recombination model from a "simple crossover" model to a "double-stranded break (DSB)" model, and configures the details of the gene conversion tracts that will therefore be modeled (see section 1.5.6 for discussion of these models). The

fraction of DSBs that will be modeled as non-crossover events is given by nonCrossoverFraction. The mean length of gene conversion tracts (whether associated with crossover or non-crossover events) is given by meanLength; the actual extent of a gene conversion tract will be the sum of two independent draws from a geometric distribution with mean meanLength/2. The fraction of gene conversion tracts that are modeled as "simple" is given by simpleConversionFraction; the remainder will be modeled as "complex", involving repair of heteroduplex mismatches. Finally, the GC bias during heteroduplex mismatch repair is given by bias, with the default of 0.0 indicating no bias, 1.0 indicating an absolute preference for G/C mutations over A/T mutations, and -1.0 indicating an absolute preference for A/T mutations over G/C mutations. A non-zero bias may only be set in nucleotide-based models (see section 1.8). This function, and the way that gene conversion is modeled, fundamentally changed in SLiM 3.3; see section 1.5.6 for discussion. Beginning in SLiM 4.1, the redrawLengthsOnFailure parameter can be used to modify the internal mechanics of layout of gene conversion tracts. If it is F (the default, and the only behavior supported before SLiM 4.1), then if an attempt to lay out gene conversion tracts fails (because the tracts overlap each other, or overlap the start or end of the chromosome), SLiM will try again by drawing new positions for the tracts - essentially shuffling the tracts around to try to find positions for them that don't overlap. If redrawLengthsOnFailure is T, then if an attempt to lay out gene conversion tracts fails, SLiM will try again by drawing new lengths for the tracts, as well as new positions. This makes it more likely that layout will succeed, but risks biasing the realized mean tract length downward from the requested mean length (since layout of long tracts is more likely fail due to overlap). In either case, if SLiM attempts to lay out gene conversion tracts 100 times without success, an error will result. That error indicates that the specified constraints for gene conversion are difficult to satisfy - tracts may commonly be so long that it is difficult or impossible to find an acceptable layout for them within the specified chromosome length. Setting redrawLengthsOnFailure to T may mitigate this problem, at the price of biasing the mean tract length downward as discussed.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Initialize: Init, initializeAncestralNucleotides(), initializeGenomicElementType(), initializeGenomicElement(), initializeHotspotMap(), initializeInteractionType(), initializeMutationRate(), initializeMutationTypeNuc(), initializeMutationType(),

`initializeRecombinationRate()`, `initializeSLiMModelType()`, `initializeSLiMOptions()`, `initializeSex()`, `initializeSpecies()`, `initializeTreeSeq()`

Examples

```
## This just brings up the documentation:
initializeGeneConversion()
```

`initializeGenomicElement`

SLiM method initializeGenomicElement

Description

Documentation for SLiM function `initializeGenomicElement`, which is a method of the SLiM class `Initialize`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
initializeGenomicElement(genomicElementType, start, end)
```

Arguments

`genomicElementType`	An object of type integer or GenomicElementType object. See details for description.
`start`	An object of type integer. See details for description.
`end`	An object of type integer. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 648.

Add a genomic element to the chromosome at initialization time. The start and end parameters give the first and last base positions to be spanned by the new genomic element. The new element will be based upon the genomic element type identified by genomicElementType, which can be either an integer, representing the ID of the desired element type, or an object of type GenomicElementType specified directly. Beginning in SLiM 3.3, this function is vectorized: the genomicElementType, start, and end parameters do not have to be singletons. In particular, start and end may be of any length, but must be equal in length; each start/end element pair will generate one new genomic element spanning the given base positions. In this case, genomicElementType may still be a singleton, providing the genomic element type to be used for all of the new genomic elements, or it may be equal in length to start and end, providing an independent genomic element type for each new element. When adding a large number of genomic elements, it will be much faster to

add them in order of ascending position with a vectorized call. The return value provides the genomic element(s) created by the call, in the order in which they were specified in the parameters to initializeGenomicElement().

Value

An object of type GenomicElement object.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Initialize: Init, initializeAncestralNucleotides(), initializeGeneConversion(), initializeGenomicElementType(), initializeHotspotMap(), initializeInteractionType(), initializeMutationRate(), initializeMutationTypeNuc(), initializeMutationType(), initializeRecombinationRate(), initializeSLiMModelType(), initializeSLiMOptions(), initializeSex(), initializeSpecies(), initializeTreeSeq()

Examples

```
## This just brings up the documentation:
initializeGenomicElement()
```

---

initializeGenomicElementType

SLiM method initializeGenomicElementType

Description

Documentation for SLiM function `initializeGenomicElementType`, which is a method of the SLiM class Initialize. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
initializeGenomicElementType(id, mutationTypes, proportions, mutationMatrix)
```

Arguments

`id`	An object of type integer or string. Must be of length 1 (a singleton). See details for description.
`mutationTypes`	An object of type integer or MutationType object. See details for description.
`proportions`	An object of type numeric. See details for description.
`mutationMatrix`	An object of type null or float. The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 648.

Add a genomic element type at initialization time. The id must not already be used for any genomic element type in the simulation. The mutationTypes vector identifies the mutation types used by the genomic element, and the proportions vector should be of equal length, specifying the relative proportion of mutations that will be drawn from the corresponding mutation type (proportions do not need to add up to one; they are interpreted relatively). The id parameter may be either an integer giving the ID of the new genomic element type, or a string giving the name of the new genomic element type (such as "g5" to specify an ID of 5). The mutationTypes parameter may be either an integer vector representing the IDs of the desired mutation types, or an object vector of MutationType elements specified directly. The global symbol for the new genomic element type is immediately available; the return value also provides the new object. The mutationMatrix parameter is NULL by default, and in non-nucleotide-based models it must be NULL. In nucleotide-based models, on the other hand, it must be non-NULL, and therefore must be supplied. In that case, mutationMatrix should take one of two standard forms. For sequence-based mutation rates that depend upon only the single nucleotide at a mutation site, mutationMatrix should be a 4×4 float matrix, specifying mutation rates for an existing nucleotide state (rows from 0- 3 representing A/C/G/T) to each of the four possible derived nucleotide states (columns, with the same meaning): The mutation rates in this matrix are absolute rates, per nucleotide per gamete; they will be used by SLiM directly unless they are multiplied by a factor from the hotspot map (see initializeHotspotMap()). Rates in mutationMatrix that involve the mutation of a nucleotide to itself (A to A, C to C, etc.) are not used by SLiM and must be 0.0 by convention (shown above with asterisks). It is important to note that the order of the rows and columns used in SLiM, A/C/G/T, is not a universal convention; other sources will present substitution-rate/transition-rate matrices using different conventions, and so care must be taken when importing such matrices into SLiM. For sequence-based mutation rates that depend upon the trinucleotide sequence centered upon a mutation site (the adjacent bases to the left and right, in other words, as well as the mutating nucleotide itself), mutationMatrix should be a 64×4 float matrix, specifying mutation rates for the central nucleotide of an existing trinucleotide sequence (rows from 0-63, representing trinucleotides as described in the documentation for the ancestralNucleotides() method of Chromosome) to each of the four possible derived nucleotide states (columns from 0-3 for A/C/G/T as before): Note that in every case it is the central nucleotide of the trinucleotide sequence that is mutating, but rates can be specified independently based upon the nucleotides in the first and third positions as well, with this type of mutation matrix. Several helper functions

are defined to construct common types of mutation matrices, such as mmJukesCantor() to create a mutation matrix for a Jukes-Cantor model; see section 25.18.1. See chapter 18 for practical examples of mutation matrices, and section 23.2.3 for further discussion of the mutational paradigm in nucleotide-based models.

Value

An object of type GenomicElementType object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Initialize: Init, initializeAncestralNucleotides(), initializeGeneConversion(), initializeGenomicElement(), initializeHotspotMap(), initializeInteractionType(), initializeMutationRate(), initializeMutationTypeNuc(), initializeMutationType(), initializeRecombinationRate(), initializeSLiMModelType(), initializeSLiMOptions(), initializeSex(), initializeSpecies(), initializeTreeSeq()

Examples

```
## This just brings up the documentation:
initializeGenomicElementType()
```

---

initializeHotspotMap *SLiM method initializeHotspotMap*

Description

Documentation for SLiM function initializeHotspotMap, which is a method of the SLiM class Initialize. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
initializeHotspotMap(multipliers, ends, sex)
```

Arguments

`multipliers`	An object of type numeric. See details for description.
`ends`	An object of type null or integer. The default value is `NULL`. See details for description.
`sex`	An object of type string. Must be of length 1 (a singleton). The default value is `"*"`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 649.

In nucleotide-based models, set the mutation rate multiplier along the chromosome. Nucleotidebased models define sequence-based mutation rates that are set up with the mutationMatrix parameter to initializeGenomicElementType(). If no hotspot map is specified by calling initializeHotspotMap(), a hotspot map with a multiplier of 1.0 across the whole chromosome is assumed (and so the sequence-based rates are the absolute mutation rates used by SLiM). A hotspot * PA→C PA→G PA→T PC→A * PC→G PC→T PG→A PG→C * PG→T PT→A PT→C PT→G * * PAAA→ACA PAAA→AGA PAAA→ATA * PAAC→ACC PAAC→AGC PAAC→ATC * PAAG→ACG PAAG→AGG PAAG→ATG * PAAT→ACT PAAT→AGT PAAT→ATT PACA→AAA * PACA→AGA PACA→ATA PACC→AAC * PACC→AGC PACC→ATC PACG→AAG * PACG→AGG PACG→ATG . . . . . PTTC→TAC PTTC→TCC PTTC→TGC * PTTG→TAG PTTG→TCG PTTG→TGG * PTTT→TAT PTTT→TCT PTTT→TGT * map modifies the sequence-based rates by scaling them up in some regions, with multipliers greater than 1.0 (representing mutational hot spots), and/or scaling them down in some regions, with multipliers less than 1.0 (representing mutational cold spots). There are two ways to call this function. If the optional ends parameter is NULL (the default), then multipliers must be a singleton value that specifies a single multiplier to be used along the entire chromosome (typically 1.0, but not required to be). If, on the other hand, ends is supplied, then multipliers and ends must be the same length, and the values in ends must be specified in ascending order. In that case, multipliers and ends taken together specify the multipliers to be used along successive contiguous stretches of the chromosome, from beginning to end; the last position specified in ends should extend to the end of the chromosome (i.e. at least to the end of the last genomic element, if not further). For example, if the following call is made: initializeHotspotMap(c(1.0, 1.2), c(5000, 9999)); then the result is that the mutation rate multiplier for bases 0...5000 (inclusive) will be 1.0 (and so the specified sequence-based mutation rates will be used verbatim), and the multiplier for bases 5001...9999 (inclusive) will be 1.2 (and so the sequence-based mutation rates will be multiplied by 1.2 within the region). Note that mutations are generated by SLiM only within genomic elements, regardless of the hotspot map. In effect, the hotspot map given is intersected with the coverage area of the genomic elements defined; areas outside of any genomic element are given a multiplier of zero. There is no harm in supplying a hotspot map that specifies multipliers for areas outside of the genomic elements defined; the excess information is simply not used. If the optional sex parameter is "*" (the default), then the supplied hotspot map will be used for both sexes (which is the only option for hermaphroditic simulations). In sexual simulations sex may be "M" or "F" instead, in which case the supplied hotspot map is used only for that sex (i.e., when generating a gamete from a parent of that sex). In this case, two calls must be made to initializeHotspotMap(), one for each sex, even if a multiplier of 1.0 is desired for the other sex; no default hotspot map is supplied.

Value

An object of type void.

Copyright

This is documentation for a function in the SLiM software, and has been reproduced from the official manual, which can be found here: http://benhaller.com/slim/SLiM_Manual.pdf. This documentation is Copyright © 2016-2020 Philipp Messer. All rights reserved. More information about SLiM can be found on the official website: https://messerlab.org/slim/

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Initialize: Init, initializeAncestralNucleotides(), initializeGeneConversion(), initializeGenomicElementType(), initializeGenomicElement(), initializeInteractionType(), initializeMutationRate(), initializeMutationTypeNuc(), initializeMutationType(), initializeRecombinationRate(), initializeSLiMModelType(), initializeSLiMOptions(), initializeSex(), initializeSpecies(), initializeTreeSeq()

Examples

```
## This just brings up the documentation:
initializeHotspotMap()
```

---

initializeInteractionType

SLiM method initializeInteractionType

Description

Documentation for SLiM function `initializeInteractionType`, which is a method of the SLiM class Initialize. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
initializeInteractionType(
  id,
  spatiality,
  reciprocal,
  maxDistance,
  sexSegregation
)
```

Arguments

`id`	An object of type integer or string. Must be of length 1 (a singleton). See details for description.
`spatiality`	An object of type string. Must be of length 1 (a singleton). See details for description.
`reciprocal`	An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.
`maxDistance`	An object of type numeric. Must be of length 1 (a singleton). The default value is `INF`. See details for description.
`sexSegregation`	An object of type string. Must be of length 1 (a singleton). The default value is `"**"`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 650.

Add an interaction type at initialization time. The id must not already be used for any interaction type in the simulation. The id parameter may be either an integer giving the ID of the new interaction type, or a string giving the name of the new interaction type (such as "i5" to specify an ID of 5). The spatiality may be "", for non-spatial interactions (i.e., interactions that do not depend upon the distance between individuals); "x", "y", or "z" for one-dimensional interactions; "xy", "xz", or "yz" for two-dimensional interactions; or "xyz" for three-dimensional interactions. The dimensions referenced by spatiality must be defined as spatial dimensions with initializeSLiMOptions(); if the simulation has dimensionality "xy", for example, then interactions in the simulation may have spatiality "", "x", "y", or "xy", but may not reference spatial dimension z and thus may not have spatiality "xz", "yz", or "xyz". If no spatial dimensions have been configured, only non-spatial interactions may be defined. The reciprocal flag may be T, in which case the interaction is guaranteed by the user to be reciprocal: whatever the interaction strength is for exerter B upon receiver A, it will be equal (in magnitude and sign) for exerter A upon receiver B. In principle, this allows the InteractionType to reduce the amount of computation necessary by up to a factor of two (although it may or may not be used). If reciprocal is F, the interaction is not guaranteed to be reciprocal and each interaction will be computed independently. The built-in interaction formulas are all reciprocal, but if you implement an interaction() callback (see section 26.7), you must consider whether the callback you have implemented preserves reciprocality or not. For this reason, the default is reciprocal=F, so that bugs are not inadvertently introduced by an invalid assumption of reciprocality. See below for a note regarding reciprocality in sexual simulations when using the sexSegregation flag. The maxDistance parameter supplies the maximum distance over which interactions of this type will be evaluated; at greater distances, the interaction strength is considered to be zero (for efficiency). The default value of maxDistance, INF (positive infinity), indicates that there is no maximum interaction distance; note that this can make some interaction queries much less efficient, and is therefore not recommended. In SLiM 3.1 and later, a warning will be issued if a spatial interaction type is defined with no maximum distance to encourage a maximum distance to be defined. The sexSegregation parameter governs the applicability of the interaction to each sex, in sexual simulations. It does not affect distance calculations in any way; it only modifies the way in which interaction strengths are calculated. The

default, "**", implies that the interaction is felt by both sexes (the first character of the string value) and is exerted by both sexes (the second character of the string value). Either or both characters may be M or F instead; for example, "MM" would indicate a male-male interaction, such as male-male competition, whereas "FM" would indicate an interaction influencing only female receivers that is influenced only by male exerters, such as male mating displays that influence female attraction. This parameter may be set only to "**" unless sex has been enabled with initializeSex(). Note that a value of sexSegregation other than "**" may imply some degree of non-reciprocality, but it is not necessary to specify reciprocal to be F for this reason; SLiM will take the sex-segregation of the interaction into account for you. The value of reciprocal may therefore be interpreted as meaning: in those cases, if any, in which A interacts with B and B interacts with A, is the interaction strength guaranteed to be the same in both directions? The sexSegregation parameter is shorthand for setting sex constraints on the interaction type using the setConstraints() method; see that method for a more extensive set of constraints that may be used. By default, the interaction strength is 1.0 for all interactions within maxDistance. Often it is desirable to change the interaction function using setInteractionFunction(); modifying interaction strengths can also be achieved with interaction() callbacks if necessary (see section 26.7). In any case, interactions beyond maxDistance always have a strength of 0.0, and the interaction strength of an individual with itself is always 0.0, regardless of the interaction function or callbacks. The global symbol for the new interaction type is immediately available; the return value also provides the new object. Note that in multispecies models, initializeInteractionType() must be called from a non-species-specific interaction() callback (declared as species all initialize()), since interactions are managed at the community level.

Value

An object of type InteractionType object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Initialize: Init, initializeAncestralNucleotides(), initializeGeneConversion(), initializeGenomicElementType(), initializeGenomicElement(), initializeHotspotMap(), initializeMutationRate(), initializeMutationTypeNuc(), initializeMutationType(), initializeRecombinationRate(), initializeSLiMModelType(), initializeSLiMOptions(), initializeSex(), initializeSpecies(), initializeTreeSeq()

Examples

```
## This just brings up the documentation:
initializeInteractionType()
```

`initializeMutationRate`

SLiM method initializeMutationRate

Description

Documentation for SLiM function `initializeMutationRate`, which is a method of the SLiM class `Initialize`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
initializeMutationRate(rates, ends, sex)
```

Arguments

`rates`	An object of type numeric. See details for description.
`ends`	An object of type null or integer. The default value is `NULL`. See details for description.
`sex`	An object of type string. Must be of length 1 (a singleton). The default value is `"*"`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 651.

Set the mutation rate per base position per gamete. To be precise, this mutation rate is the expected mean number of mutations that will occur per base position per gamete; note that this is different from how the recombination rate is defined (see initializeRecombinationRate()). The number of mutations that actually occurs at a given base position when generating an offspring genome is, in effect, drawn from a Poisson distribution with that expected mean (but under the hood SLiM uses a mathematically equivalent but much more efficient strategy). It is possible for this Poisson draw to indicate that two or more new mutations have arisen at the same base position, particularly when the mutation rate is very high; in this case, the new mutations will be added to the site one at a time, and as always the mutation stacking policy (see section 1.5.3) will be followed. There are two ways to call this function. If the optional ends parameter is NULL (the default), then rates must be a singleton value that specifies a single mutation rate to be used along the entire chromosome. If, on the other hand, ends is supplied, then rates and ends must be the same length, and the values in ends must be specified in ascending order. In that case, rates and ends taken together specify the mutation rates to be used along successive contiguous stretches

of the chromosome, from beginning to end; the last position specified in ends should extend to the end of the chromosome (i.e. at least to the end of the last genomic element, if not further). For example, if the following call is made: initializeMutationRate(c(1e-7, 2.5e-8), c(5000, 9999)); then the result is that the mutation rate for bases 0...5000 (inclusive) will be 1e-7, and the rate for bases 5001...9999 (inclusive) will be 2.5e-8. Note that mutations are generated by SLiM only within genomic elements, regardless of the mutation rate map. In effect, the mutation rate map given is intersected with the coverage area of the genomic elements defined; areas outside of any genomic element are given a mutation rate of zero. There is no harm in supplying a mutation rate map that specifies rates for areas outside of the genomic elements defined; that rate information is simply not used. The overallMutationRate family of properties on Chromosome provide the overall mutation rate after genomic element coverage has been taken into account, so it will reflect the rate at which new mutations will actually be generated in the simulation as configured. If the optional sex parameter is "*" (the default), then the supplied mutation rate map will be used for both sexes (which is the only option for hermaphroditic simulations). In sexual simulations sex may be "M" or "F" instead, in which case the supplied mutation rate map is used only for that sex (i.e., when generating a gamete from a parent of that sex). In this case, two calls must be made to initializeMutationRate(), one for each sex, even if a rate of zero is desired for the other sex; no default mutation rate map is supplied. In nucleotide-based models, initializeMutationRate() may not be called. Instead, the desired sequence-based mutation rate(s) should be expressed in the mutationMatrix parameter to initializeGenomicElementType(). If variation in the mutation rate along the chromosome is desired, initializeHotspotMap() should be used.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Initialize: Init, initializeAncestralNucleotides(), initializeGeneConversion(), initializeGenomicElementType(), initializeGenomicElement(), initializeHotspotMap(), initializeInteractionType(), initializeMutationTypeNuc(), initializeMutationType(), initializeRecombinationRate(), initializeSLiMModelType(), initializeSLiMOptions(), initializeSex(), initializeSpecies(), initializeTreeSeq()

Examples

```
## This just brings up the documentation:
initializeMutationRate()
```

`initializeMutationType`

SLiM method initializeMutationType

Description

Documentation for SLiM function `initializeMutationType`, which is a method of the SLiM class `Initialize`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
initializeMutationType(id, dominanceCoeff, distributionType, ...)
```

Arguments

`id`	An object of type integer or string. Must be of length 1 (a singleton). See details for description.
`dominanceCoeff`	An object of type numeric. Must be of length 1 (a singleton). See details for description.
`distributionType`	An object of type string. Must be of length 1 (a singleton). See details for description.
`...`	An object of type NA. NA See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 652.

Add a mutation type at initialization time. The id must not already be used for any mutation type in the simulation. The id parameter may be either an integer giving the ID of the new mutation type, or a string giving the name of the new mutation type (such as "m5" to specify an ID of 5). The dominanceCoeff parameter supplies the dominance coefficient for the mutation type; 0.0 produces no dominance, 1.0 complete dominance, and values greater than 1.0, overdominance. The distributionType may be "f", in which case the ellipsis ... should supply a numeric$ fixed selection coefficient; "e", in which case the ellipsis should supply a numeric$ mean selection coefficient for an exponential distribution; "g", in which case the ellipsis should supply a numeric$ mean selection coefficient and a numeric$ alpha shape parameter for a gamma distribution; "n", in which case the ellipsis should supply a numeric$ mean selection coefficient and a numeric$ sigma (standard deviation)

parameter for a normal distribution; "p", in which case the ellipsis should supply a numeric$ mean selection coefficient and a numeric$ scale parameter for a Laplace distribution; "w", in which case the ellipsis should supply a numeric$ scale parameter and a numeric$ k shape parameter for a Weibull distribution; or "s", in which case the ellipsis should supply a string$ Eidos script parameter. See section 25.11 for discussion of the various DFEs and their uses. The global symbol for the new mutation type is immediately available; the return value also provides the new object. Note that by default in WF models, all mutations of a given mutation type will be converted into Substitution objects when they reach fixation, for efficiency reasons. If you need to disable this conversion, to keep mutations of a given type active in the simulation even after they have fixed, you can do so by setting the convertToSubstitution property of MutationType to F. In contrast, by default in nonWF models mutations will not be converted into Substitution objects when they reach fixation; convertToSubstitution is F by default in nonWF models. To enable conversion in nonWF models for neutral mutation types with no indirect fitness effects, you should therefore set convertToSubstitution to T. See sections 23.3, 24.5, and 25.11.1 for further discussion regarding the convertToSubstitution property.

Value

An object of type MutationType object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Initialize: Init, initializeAncestralNucleotides(), initializeGeneConversion(), initializeGenomicElementType(), initializeGenomicElement(), initializeHotspotMap(), initializeInteractionType(), initializeMutationRate(), initializeMutationTypeNuc(), initializeRecombinationRate(), initializeSLiMModelType(), initializeSLiMOptions(), initializeSex(), initializeSpecies(), initializeTreeSeq()

Examples

```
## This just brings up the documentation:
initializeMutationType()
```

`initializeMutationTypeNuc`
SLiM method initializeMutationTypeNuc

Description

Documentation for SLiM function `initializeMutationTypeNuc`, which is a method of the SLiM class `Initialize`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
initializeMutationTypeNuc(id, dominanceCoeff, distributionType, ...)
```

Arguments

`id`	An object of type integer or string. Must be of length 1 (a singleton). See details for description.
`dominanceCoeff`	An object of type numeric. Must be of length 1 (a singleton). See details for description.
`distributionType`	An object of type string. Must be of length 1 (a singleton). See details for description.
`...`	An object of type NA. NA See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 653.

Add a nucleotide-based mutation type at initialization time. This function is identical to initializeMutationType() except that the new mutation type will be nucleotide-based - in other words, mutations belonging to the new mutation type will have an associated nucleotide. This function may be called only in nucleotide-based models (as enabled by the nucleotideBased parameter to initializeSLiMOptions()). Nucleotide-based mutations always use a mutationStackGroup of -1 and a mutationStackPolicy of "l". This ensures that a new nucleotide mutation always replaces any previously existing nucleotide mutation at a given position, regardless of the mutation types of the nucleotide mutations. These values are set automatically by initializeMutationTypeNuc(), and may not be changed. See the documentation for initializeMutationType() for all other discussion.

Value

An object of type MutationType object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Initialize: Init, initializeAncestralNucleotides(), initializeGeneConversion(), initializeGenomicElementType(), initializeGenomicElement(), initializeHotspotMap(), initializeInteractionType(), initializeMutationRate(), initializeMutationType(), initializeRecombinationRate(), initializeSLiMModelType(), initializeSLiMOptions(), initializeSex(), initializeSpecies(), initializeTreeSeq()

Examples

```
## This just brings up the documentation:
initializeMutationTypeNuc()
```

---

initializeRecombinationRate

SLiM method initializeRecombinationRate

Description

Documentation for SLiM function `initializeRecombinationRate`, which is a method of the SLiM class `Initialize`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
initializeRecombinationRate(rates, ends, sex)
```

Arguments

`rates`	An object of type numeric. See details for description.
`ends`	An object of type null or integer. The default value is `NULL`. See details for description.
`sex`	An object of type string. Must be of length 1 (a singleton). The default value is `"*"`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 653.

Set the recombination rate per base position per gamete. To be precise, this recombination rate is the probability that a breakpoint will occur between one base and the next base; note that this is different from how the mutation rate is defined (see initializeMutationRate()). All rates must be in the interval [0.0, 0.5]. A rate of 0.5 implies complete independence between the adjacent bases, which might be used to implement independent assortment of loci located on different chromosomes (see the example below). Whether a breakpoint occurs between two bases is then, in effect, determined by a binomial draw with a single trial and the given rate as probability (but under the hood SLiM uses a mathematically equivalent but much more efficient strategy). The recombinational process in SLiM will never generate more then one crossover between one base and the next (in one generation/ genome), and a supplied rate of 0.5 will therefore result in an actual probability of 0.5 for a crossover at the relevant position. (Note that this was not true in SLiM 2.x and earlier, however; their implementation of recombination resulted in a crossover probability of about 39.3 inaccurate approximation method. Recombination rates lower than about 0.01 would have been essentially exact, since the approximation error became large only as the rate approached 0.5.) There are two ways to call this function. If the optional ends parameter is NULL (the default), then rates must be a singleton value that specifies a single recombination rate to be used along the entire chromosome. If, on the other hand, ends is supplied, then rates and ends must be the same length, and the values in ends must be specified in ascending order. In that case, rates and ends taken together specify the recombination rates to be used along successive contiguous stretches of the chromosome, from beginning to end; the last position specified in ends should extend to the end of the chromosome (i.e. at least to the end of the last genomic element, if not further). Note that a recombination rate of 1 centimorgan/Mbp corresponds to a recombination rate of 1e-8 in the units used by SLiM. For example, if the following call is made: initializeRecombinationRate(c(0, 0.5, 0), c(5000, 5001, 9999)); then the result is that the recombination rates between bases 0 / 1, 1 / 2, ..., 4999 / 5000 will be 0, the rate between bases 5000 / 5001 will be 0.5, and the rate between bases 5001 / 5002 onward (up to 9998 / 9999) will again be 0. Setting the recombination rate between one specific pair of bases to 0.5 forces recombination to occur with a probability of 0.5 between those bases, which effectively breaks the simulated locus into separate chromosomes at that point; this example effectively has one simulated chromosome from base position 0 to 5000, and another from 5001 to 9999. If the optional sex parameter is "*" (the default), then the supplied recombination rate map will be used for both sexes (which is the only option for hermaphroditic simulations). In sexual simulations sex may be "M" or "F" instead, in which case the supplied recombination map is used only for that sex. In this case, two calls must be made to initializeRecombinationRate(), one for each sex, even if a rate of zero is desired for the other sex; no default recombination map is supplied.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Initialize: Init, initializeAncestralNucleotides(), initializeGeneConversion(), initializeGenomicElementType(), initializeGenomicElement(), initializeHotspotMap(), initializeInteractionType(), initializeMutationRate(), initializeMutationTypeNuc(), initializeMutationType(), initializeSLiMModelType(), initializeSLiMOptions(), initializeSex(), initializeSpecies(), initializeTreeSeq()

Examples

```
## This just brings up the documentation:
initializeRecombinationRate()
```

---

initializeSex *SLiM method initializeSex*

Description

Documentation for SLiM function `initializeSex`, which is a method of the SLiM class `Initialize`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
initializeSex(chromosomeType)
```

Arguments

`chromosomeType`
: An object of type string. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 654.

Enable and configure sex in the simulation. The argument chromosomeType gives the type of chromosome to be simulated; this should be "A", "X", or "Y". Calling this function has the side effect of enabling sex in the simulation; individuals will be male and female (rather than hermaphroditic) regardless of the chromosomeType chosen for simulation. There is no way to disable sex once it has been enabled; if you don't want to have sex, don't call this function. The xDominanceCoeff parameter has been deprecated and removed. In SLiM 3.7 and later, use the haploidDominanceCoeff property of MutationType instead. If the chromosomeType is "X", the optional xDominanceCoeff parameter can supply the dominance coefficient used when a mutation is present in an XY male, and is thus "heterozygous" (but in a different sense than the heterozygosity of an XX female with one copy of the mutation).

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Initialize: Init, initializeAncestralNucleotides(), initializeGeneConversion(), initializeGenomicElementType(), initializeGenomicElement(), initializeHotspotMap(), initializeInteractionType(), initializeMutationRate(), initializeMutationTypeNuc(), initializeMutationType(), initializeRecombinationRate(), initializeSLiMModelType(), initializeSLiMOptions(), initializeSpecies(), initializeTreeSeq()

Examples

```
## This just brings up the documentation:
initializeSex()
```

`initializeSLiMModelType`
SLiM method initializeSLiMModelType

Description

Documentation for SLiM function `initializeSLiMModelType`, which is a method of the SLiM class `Initialize`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
initializeSLiMModelType(modelType)
```

Arguments

`modelType`	An object of type string. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 654.

Configure the type of SLiM model used for the simulation. At present, one of two model types may be selected. If modelType is "WF", SLiM will use a Wright-Fisher (WF) model; this is the model type that has always been supported by SLiM, and is the model type used if initializeSLiMModelType() is not called. If modelType is "nonWF", SLiM will use a non-Wright-Fisher (nonWF) model instead; this is a new model type supported by SLiM 3.0 and above (see section 1.6). If initializeSLiMModelType() is called at all then it must be called before any other initialization function, so that SLiM knows from the outset which features are enabled and which are not.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Initialize: `Init`, `initializeAncestralNucleotides()`, `initializeGeneConversion()`, `initializeGenomicElementType()`, `initializeGenomicElement()`, `initializeHotspotMap()`, `initializeInteractionType()`, `initializeMutationRate()`, `initializeMutationTypeNuc()`, `initializeMutationType()`, `initializeRecombinationRate()`, `initializeSLiMOptions()`, `initializeSex()`, `initializeSpecies()`, `initializeTreeSeq()`

Examples

```
## This just brings up the documentation:
initializeSLiMModelType()
```

`initializeSLiMOptions`

SLiM method initializeSLiMOptions

Description

Documentation for SLiM function `initializeSLiMOptions`, which is a method of the SLiM class `Initialize`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
initializeSLiMOptions(
  keepPedigrees,
  dimensionality,
  periodicity,
  mutationRuns,
  preventIncidentalSelfing,
  nucleotideBased,
  randomizeCallbacks
)
```

Arguments

`keepPedigrees`	An object of type logical or string or string or integer or logical or logical or logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.
`dimensionality`	An object of type logical or string or string or integer or logical or logical or logical. Must be of length 1 (a singleton). The default value is `""`. See details for description.

`periodicity` An object of type logical or string or string or integer or logical or logical or logical. Must be of length 1 (a singleton). The default value is `""`. See details for description.

`mutationRuns` An object of type logical or string or string or integer or logical or logical or logical. Must be of length 1 (a singleton). The default value is `0`. See details for description.

`preventIncidentalSelfing`
An object of type logical or string or string or integer or logical or logical or logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

`nucleotideBased`
An object of type logical or string or string or integer or logical or logical or logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

`randomizeCallbacks`
An object of type logical or string or string or integer or logical or logical or logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 654.

Configure options for the simulation. If initializeSLiMOptions() is called at all then it must be called before any other initialization function (except initializeSLiMModelType()), so that SLiM knows from the outset which optional features are enabled and which are not. If keepPedigrees is T, SLiM will keep pedigree information for every individual in the simulation, tracking the identity of its parents and grandparents. This allows individuals to assess their degree of pedigree-based relatedness to other individuals (see Individual's relatedness() and sharedParentCount() methods, section 25.7.2), as well as allowing a model to find "trios" (two parents and an offspring they generated) using the pedigree properties of Individual (section 25.7.1). As a side effect of keepPedigrees being T, the pedigreeID, pedigreeParentIDs, and pedigreeGrandparentIDs properties of Individual will have defined values (see section 25.7.1), as will the genomePedigreeID property of Genome (see section 25.4.1). Note that pedigree-based relatedness doesn't necessarily correspond to genetic relatedness, due to effects such as assortment and recombination. For an overview of other ways of tracking genetic ancestry, including true local ancestry at each position on the chromosome, see sections 1.7 and 14.7. Beginning in SLiM 3.5, keepPedigrees=T also enables tracking of individual reproductive output, available through the reproductiveOutput property of Individual (see section 25.7.1) and the lifetimeReproductiveOutput property of Subpopulation (see section 25.16.1). If dimensionality is not "", SLiM will enable its optional "continuous space" facility. Three values for dimensionality are presently supported: "x", "xy", and "xyz", specifying that continuous space should be enabled for one, two, or three dimensions, respectively, using (x), (x, y), and (x, y, z) coordinates respectively. This has a number of side effects. First of all, it means that the specified properties of Individual (x, y, and/or z) will be interpreted by SLiM as spatial positions; in particular, SLiMgui will use those properties to display subpopulations spatially. Second, it allows spatial interactions to be defined, evaluated, and queried using initializeInteractionType()

and interaction() callbacks. And third, it enables the use of any other properties and methods related to continuous space, such as setting the spatial boundaries of subpopulations, which would otherwise raise an error. If periodicity is not "", SLiM will designate the specified spatial dimensions as being periodic - wrapping around at the edges of the spatial boundaries of that dimension. This option may only be used if the dimensionality parameter to initializeSLiMOptions() has been used to enable spatiality in the model, and only spatial dimensions that were specified in the dimensionality of the model may be declared to be periodic (but if desired, it is permissible to make just a subset of those dimensions periodic; it is not an all-or-none proposition). For example, if the specified dimensionality is "xy", the model's periodicity may be "x", "y", or "xy" (or "", the default, to specify that there are no periodic dimensions). A one-dimensional periodic model would model a space like the perimeter of a circle. A two-dimensional model periodic in one of those dimensions would model a space like a cylinder without its end caps; if periodic in both dimensions, the modeled space is a torus. The shapes of three-dimensional periodic models are harder to visualize, but are essentially higherdimensional analogues of these concepts. Periodic boundary conditions are commonly used to model spatial scenarios without "edge effects", since there are no edges in the periodic spatial dimensions. The pointPeriodic() method of Subpopulation is typically used in conjunction with this option, to actually implement the periodic boundary condition for the specified dimensions. If mutationRuns is not 0, SLiM will use the value given as the number of mutation runs inside Genome objects; if it is 0 (the default), SLiM will calculate a number of mutation runs that it estimates will work well. Internally, SLiM divides genomes into a sequence of consecutive mutation runs, allowing more efficient internal computations. The optimal mutation run length is short enough that each mutation run is relatively unlikely to be modified by mutation/recombination events when inherited, but long enough that each mutation run is likely to contain a relatively large number of mutations; these priorities are in tension, so an intermediate balance between them is generally desirable. The optimal number of mutation runs will depend upon the machine and even the compiler used to build SLiM, so SLiM's default value may not be optimal; for maximal performance it can thus be beneficial to experiment with different values and find the optimal value for the simulation - a process which SLiM can assist with (see section 21.4). Specifying the number of mutation runs is an advanced technique, but in certain cases it can improve performance significantly. If preventIncidentalSelfing is T, incidental selfing in hermaphroditic models will be prevented by SLiM. By default (i.e., if preventIncidentalSelfing is F), SLiM chooses the first and second parents in a biparental mating event independently. It is therefore possible for the same individual to be chosen as both the first and second parent, resulting in selfing events even when the selfing rate is zero. In many models this is unimportant, since it happens fairly infrequently and does not have large consequences. This behavior is SLiM's default because it is the simplest option, and produces results that most closely align with simple analytical population genetics models. However, in some models this selfing can be undesirable and problematic. In particular, models that involve very high variance in fitness or very small effective population sizes may see elevated rates of selfing that substantially influence model results. If preventIncidentalSelfing is set to T, all such incidental selfing will be prevented (by choosing a new second parent if the first parent was chosen again). Non-incidental selfing, as requested by the selfing rate, will still be permitted. Note that if incidental selfing is prevented, SLiM will hang if it is unable to find a different second parent; there must always be at least two individuals in the population with non-zero fitness, and mateChoice() and modifyChild() callbacks must not absolutely prevent those two individuals from producing viable offspring.

Enforcement of the prohibition on incidental selfing will occur after mateChoice() callbacks have been called (and thus the default mating weights provided to mateChoice() callbacks will not exclude the first parent!), but will occur before modifyChild() callbacks are called (so those callbacks may assume that the first and second parents are distinct). If nucleotideBased is T, the model will be nucleotide-based. In this case, auto-generated mutations (i.e., mutation types used by genomic element types) must be nucleotide-based, and an ancestral nucleotide sequence must be supplied with initializeAncestralNucleotides(). Nonnucleotide-based mutations may still be used, but may not be referenced by genomic element types. A mutation rate (or rate map) may not be supplied with initializeMutationRate(); instead, a hotspot map may (optionally) be supplied with initializeHotspotMap(). This choice has many consequences across SLiM; see section 1.8 for further discussion. If randomizeCallbacks is T (the default), the order in which individuals are processed in callbacks will be randomized to make it easier to avoid order-dependency bugs. This flag exists because the order of individuals in each subpopulation is non-random; most notably, females always come before males in the individuals vector, but non-random ordering may also occur with respect to things like migrant versus non-migrant status, origin by selfing versus cloning versus biparental mating, and other factors. When this option is F, individuals in a subpopulation are processed in the order of the individuals vector in each tick cycle stage, which may lead to order-dependency issues if there is an enabled callback whose behavior is not fully independent between calls. Setting this option to T will cause individuals within each subpopulation to be processed in a randomized order in each tick cycle stage; specifically, this randomizes the order of calls to mutationEffect() callbacks in both WF and nonWF models, and calls to reproduction() and survival() callbacks in nonWF models. Each subpopulation is still processed separately, in sequential order, so order-dependency issues between subpopulations are still possible if callbacks have effects that are not fully independent. This feature was added in SLiM 4, breaking backward compatibility; to recover the behavior of previous versions of SLiM, pass F for this option (but then be very careful about order-dependency issues in your script). The default of T is the safe option, but a small speed penalty is incurred by the randomization of the processing order - for most models the difference will be less than 1 the worst case it may approach 10 issue may therefore run somewhat faster if this is set to F. Note that anywhere that your script uses the individuals property of Subpopulation, the order of individuals returned will be non-random (regardless of the setting of this option); you should use sample() to shuffle the order of the individuals vector if necessary to avoid order-dependency issues in your script. This function will likely be extended with further options in the future, added on to the end of the argument list. Using named arguments with this call is recommended for readability. Note that turning on optional features may increase the runtime and memory footprint of SLiM.

Value

An object of type void.

Copyright

org/slim/

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Initialize: Init, initializeAncestralNucleotides(), initializeGeneConversion(), initializeGenomicElementType(), initializeGenomicElement(), initializeHotspotMap(), initializeInteractionType(), initializeMutationRate(), initializeMutationTypeNuc(), initializeMutationType(), initializeRecombinationRate(), initializeSLiMModelType(), initializeSex(), initializeSpecies(), initializeTreeSeq()

Examples

```
## This just brings up the documentation:
initializeSLiMOptions()
```

---

initializeSpecies *SLiM method initializeSpecies*

Description

Documentation for SLiM function `initializeSpecies`, which is a method of the SLiM class Initialize. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
initializeSpecies(tickModulo, tickPhase, avatar, color)
```

Arguments

tickModulo	An object of type integer or integer or string or string. Must be of length 1 (a singleton). The default value is 1. See details for description.
tickPhase	An object of type integer or integer or string or string. Must be of length 1 (a singleton). The default value is 1. See details for description.
avatar	An object of type integer or integer or string or string. Must be of length 1 (a singleton). The default value is "". See details for description.
color	An object of type integer or integer or string or string. Must be of length 1 (a singleton). The default value is "". See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 656.

Configure options for the species being initialized. This initialization function may only be called in multispecies models (i.e., models with explicit species declarations); in single-species models, the default values are assumed and cannot be changed. The tickModulo and tickPhase parameters determine the activation schedule for the species. The active property of the species will be set to T (thus activating the species) every tickModulo ticks, beginning in tick tickPhase. (However, when the species is activated in a given tick, the skipTick() method may still be called in a first() event to deactivate it.) See the active property of Species (section 25.15.1) for more details. The avatar parameter, if not "", sets a string value used to represent the species graphically, particularly in SLiMgui but perhaps in other contexts also. The avatar should generally be a single character - usually an emoji corresponding to the species, such as " " for foxes or " " for mice. If avatar is the empty string, "", SLiMgui will choose a default avatar. The color parameter, if not "", sets a string color value used to represent the species in SLiMgui. Colors may be specified by name, or with hexadecimal RGB values of the form "#RRGGBB" (see the Eidos manual for details). If color is the empty string, "", SLiMgui will choose a default color.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Initialize: Init, initializeAncestralNucleotides(), initializeGeneConversion(), initializeGenomicElementType(), initializeGenomicElement(), initializeHotspotMap(), initializeInteractionType(), initializeMutationRate(), initializeMutationTypeNuc(), initializeMutationType(), initializeRecombinationRate(), initializeSLiMModelType(), initializeSLiMOptions(), initializeSex(), initializeTreeSeq()

Examples

```
## This just brings up the documentation:
initializeSpecies()
```

`initializeTreeSeq` *SLiM method initializeTreeSeq*

Description

Documentation for SLiM function `initializeTreeSeq`, which is a method of the SLiM class `Initialize`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
initializeTreeSeq(
  recordMutations,
  simplificationRatio,
  simplificationInterval,
  checkCoalescence,
  runCrosschecks,
  retainCoalescentOnly,
  timeUnit
)
```

Arguments

`recordMutations`
: An object of type logical. Must be of length 1 (a singleton). The default value is `T`. See details for description.

`simplificationRatio`
: An object of type null or integer or float. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

`simplificationInterval`
: An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

`checkCoalescence`
: An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

`runCrosschecks`
: An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

`retainCoalescentOnly`
: An object of type logical. Must be of length 1 (a singleton). The default value is `T`. See details for description.

`timeUnit`
: An object of type null or string. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 656.

Configure options for tree sequence recording. Calling this function turns on tree sequence recording, as a side effect, for later reconstruction of the simulation's evolutionary dynamics; if you do not want tree sequence recording to be enabled, do not call this function. Note that tree-sequence recording internally uses SLiM's "pedigree tracking" feature to uniquely identify individuals and genomes; however, if you want to use pedigree tracking in your script you must still enable it yourself with initializeSLiMOptions(keepPedigrees=T). The recordMutations flag controls whether information about individual mutations is recorded or not. Such recording takes time and memory, and so can be turned off if only the tree sequence itself is needed, but it is turned on by default since mutation recording is generally useful. The simplificationRatio and simplificationInterval parameters control how often automatic simplification of the recorded tree sequence occurs. This is a speed-memory tradeoff: more frequent simplification (lower simplificationRatio or smaller simplificationInterval) means the stored tree sequences will use less memory, but at a cost of somewhat longer run times. Conversely, a larger simplificationRatio or simplificationInterval means that SLiM will wait longer between simplifications. There are three ways these parameters can be used. With the first option, with a non- NULL simplificationRatio and a NULL value for simplificationInterval, SLiM will try to find an optimal tick interval for simplification such that the ratio of the memory used by the tree sequence tables, (before:after) simplification, is close to the requested ratio. The default of 10 (used if both simplificationRatio and simplificationInterval are NULL) thus requests that SLiM try to find a tick interval such that the maximum size of the stored tree sequences is ten times the size after simplification. INF may be supplied to indicate that automatic simplification should never occur; 0 may be supplied to indicate that automatic simplification should be performed at the end of every tick. Alternatively - the second option - simplificationRatio may be NULL and simplificationInterval may be set to the interval, in ticks, between simplifications. This may provide more reliable performance, but the interval must be chosen carefully to avoid exceeding the available memory. The simplificationInterval value may be a very large number to specify that simplification should never occur (not INF, though, since it is an integer value), or 1 to simplify every tick. Finally - the third option - both parameters may be non-NULL, in which case simplificationRatio is used as described above, while simplificationInterval provides the initial interval first used by SLiM (and then subsequently increased or decreased to try to match the requested simplification ratio). The default initial interval, used when simplificationInterval is NULL, is usually 20; this is chosen to be relatively frequent, and thus unlikely to lead to a memory overflow, but it can result in rather slow spool-up for models where the equilibrium simplification interval, as determined by the simplification ratio, is much longer. It can therefore be helpful to set a larger initial interval so that the early part of the model run is not excessively bogged down in simplification. The checkCoalescence parameter controls whether a check for full coalescence is conducted after each simplification. If a model will call treeSeqCoalesced() to check for coalescence during its execution, checkCoalescence should be set to T. Since the coalescence checks entail a performance penalty, the default of F is preferable otherwise. See the documentation for treeSeqCoalesced() for further discussion. The runCrosschecks parameter controls whether cross-checks between SLiM's internal data structures and the tree-sequence recording data structures will be conducted. These two sets of data structures record much the same thing (mutations in genomes), but using completely different representations, so such cross-checks can be useful to confirm that the two data structures

do indeed represent the same conceptual state. This slows down the model considerably, however, and would normally be turned on only for debugging purposes, so it is turned off by default. The retainCoalescentOnly parameter controls how, exactly, simplification of the tree-sequence data is performed in SLiM (both for auto-simplification and for calls to treeSeqSimplify()). More specifically, this parameter controls the behavior of simplification for individuals and genomes that have been "retained" by calling treeSeqRememberIndividuals() with the parameter permanent=F. The default of retainCoalescentOnly=T helps to keep the number of retained individuals relatively small, which is helpful if your simulation regularly flags many individuals for retaining. In this case, changing retainCoalescentOnly to F may dramatically increase memory usage and runtime, in a similar way to permanently remembering all the individuals. See the documentation of treeSeqRememberIndividuals() for further discussion (section 25.15.2). The timeUnit parameter controls the time unit stated in the tree sequence when it is saved (which can be accessed through tskit APIs); it has no effect on the running simulation whatsoever. The default value, NULL, means that a time unit of "ticks" will be used for all model types. (In SLiM 3.7 / 3.7.1, NULL implied a time unit of "generations" for WF models, but "ticks" for nonWF models; given the new multispecies timescale parameters in SLiM 4, a default of "ticks" makes sense in all cases since now even in WF models one tick might not equal one biological generation.) It may be helpful to set timeUnit to "generations" explicitly when modeling non-overlapping generations in which one tick equals one generation, to tell tskit that the time unit does in fact represent biological generations; doing so may avoid warnings from tskit or msprime regarding the time unit, in cases such as recapitation where the simulation timescale is important.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Initialize: Init, initializeAncestralNucleotides(), initializeGeneConversion(), initializeGenomicElementType(), initializeGenomicElement(), initializeHotspotMap(), initializeInteractionType(), initializeMutationRate(), initializeMutationTypeNuc(), initializeMutationType(), initializeRecombinationRate(), initializeSLiMModelType(), initializeSLiMOptions(), initializeSex(), initializeSpecies()

Examples

```
## This just brings up the documentation:
initializeTreeSeq()
```

`interactingNeighborCount`

SLiM method interactingNeighborCount

Description

Documentation for SLiM function `interactingNeighborCount`, which is a method of the SLiM class `InteractionType`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
interactingNeighborCount(receivers, exerterSubpop)
```

Arguments

`receivers`	An object of type Individual object. See details for description.
`exerterSubpop`	An object of type null or Subpopulation object. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 693.

Returns the number of interacting individuals for each individual in receivers, within the maximum interaction distance according to the distance metric of the InteractionType, from among the exerters in exerterSubpop (or, if that is NULL, then from among all individuals in the receiver's subpopulation). More specifically, this method counts the number of individuals which can exert an interaction upon each receiver (which does not include the receiver itself). All of the receivers must belong to a single subpopulation, and all of the exerters must belong to a single subpopulation, but those two subpopulations do not need to be the same. The evaluate() method must have been previously called for the receiver and exerter subpopulations, and positions saved at evaluation time will be used. This method is similar to nearestInteractingNeighbors() (when passed a large count so as to guarantee that all interacting individuals are returned), but this method returns only a count of the interacting individuals, not a vector containing the individuals. Note that this method uses interaction eligibility as a criterion; it will not count neighbors that do not exert an interaction upon a given receiver (due to the configured receiver or exerter constraints). (It also does not count a receiver as a neighbor of itself.) If a count of all neighbors is desired, rather than just interacting neighbors, use neighborCount(). If the InteractionType is non-spatial, this method may not be called.

Value

An object of type integer.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other InteractionType: `IT`, `clippedIntegral()`, `distanceFromPoint()`, `distance()`, `drawByStrength()`, `evaluate()`, `interactionDistance()`, `localPopulationDensity()`, `nearestInteractingNeighbors()`, `nearestNeighborsOfPoint()`, `nearestNeighbors()`, `neighborCountOfPoint()`, `neighborCount()`, `setConstraints()`, `setInteractionFunction()`, `strength()`, `testConstraints()`, `totalOfNeighborStrengths`
`unevaluate()`

`interaction` *SLiM interaction() callback*

Description

This callback specifies that a code block is providing logic to determine the strength of interaction between individuals. It should return the strength of the interaction as a numeric value. You must explicitly use `return(value)` at the end of the code block. For more information on how to use `interaction()` callback see SLiM Manual: page 604

Usage

```
interaction(int_type_id, subpop_id)
```

Arguments

`int_type_id`	The id of the InteractionType to apply this callback to. Can be an integer 1, 2, etc., or character "i1", "i2", etc.
`subpop_id`	The id(s) of the subpopulation(s) to which this callback should apply. Can be an integer 1, 2, etc., or character "p1", "p2", etc.

Details

Global variables available in reproduction callbacks:

distance The distance from receiver to exerter, in spatial simulations; NAN otherwise

strength The default interaction strength calculated by the interaction function

receiver The individual receiving the interaction (an object of class Individual)

exerter The individual exerting the interaction (an object of class Individual)

subpop The subpopulation in which the receiver and exerter live

Value

None

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other callbacks: early(), first(), fitnessEffect(), fitness(), initialize(), late(), mateChoice(), modifyChild(), mutationEffect(), mutation(), recombination(), reproduction(), slim_callbacks(), survival()

Examples

```
slim_block(interaction(), {
  # custom interaction strength as a function of values held in exerter and receiver tagF element.
  return(dnorm(exerter.tagF, receiver.tagF, 0.1) / dnorm(0, 0, 0.1))
})
```

---

interactionDistance *SLiM method interactionDistance*

Description

Documentation for SLiM function interactionDistance, which is a method of the SLiM class InteractionType. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
interactionDistance(receiver, exerters)
```

Arguments

`receiver`	An object of type Individual object. Must be of length 1 (a singleton). See details for description.
`exerters`	An object of type null or Individual object. The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 694.

Returns a vector containing interaction-dependent distances between receiver and individuals in exerters. If exerters is NULL (the default), then a vector of the interaction-dependent distances from receiver to all individuals in its subpopulation (including receiver itself) is returned; this case may be handled much more efficiently than if a vector of all individuals in the subpopulation is explicitly provided. Otherwise, all individuals in exerters must belong to a single subpopulation (but not necessarily the same subpopulation as receiver). The evaluate() method must have been previously called for the receiver and exerter subpopulations, and positions saved at evaluation time will be used. If the InteractionType is non-spatial, this method may not be called. Importantly, distances are calculated according to the spatiality of the InteractionType (as declared in initializeInteractionType()), not the dimensionality of the model as a whole (as declared in initializeSLiMOptions()). The distances returned are therefore the distances that would be used to calculate interaction strengths. In addition, interactionDistance() will return INF as the distance between receiver and any individual which does not exert an interaction upon receiver; the interactionDistance() between an individual and itself will thus be INF, and likewise for pairs excluded from interacting by receiver constraints, exerter constraints, or the maximum interaction distance of the interaction type. See distance() for an alternative distance definition.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other InteractionType: IT, clippedIntegral(), distanceFromPoint(), distance(), drawByStrength(), evaluate(), interactingNeighborCount(), localPopulationDensity(), nearestInteractingNeighbors(), nearestNeighborsOfPoint(), nearestNeighbors(), neighborCountOfPoint(), neighborCount(), setConstraints(), setInteractionFunction(), strength(), testConstraints(), totalOfNeighborStrengths(), unevaluate()

`interactionTypesWithIDs`

SLiM method interactionTypesWithIDs

Description

Documentation for SLiM function `interactionTypesWithIDs`, which is a method of the SLiM class `Community`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
interactionTypesWithIDs(ids)
```

Arguments

`ids`	An object of type integer. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 666.

Find and return the InteractionType objects with id values matching the values in ids. If no matching InteractionType object can be found with a given id, an error results.

Value

An object of type InteractionType object.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Community: `Co`, `createLogFile()`, `deregisterScriptBlock()`, `genomicElementTypesWithIDs()`, `mutationTypesWithIDs()`, `outputUsage()`, `registerEarlyEvent()`, `registerFirstEvent()`, `registerInteractionCallback()`, `registerLateEvent()`, `rescheduleScriptBlock()`, `scriptBlocksWithIDs()`, `simulationFinished()`, `speciesWithIDs()`, `subpopulationsWithIDs()`, `usage()`

`interpolate` *SLiM method interpolate*

Description

Documentation for SLiM function `interpolate`, which is a method of the SLiM class `SpatialMap`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
interpolate(factor, method)
```

Arguments

`factor`	An object of type integer or string. Must be of length 1 (a singleton). See details for description.
`method`	An object of type integer or string. Must be of length 1 (a singleton). The default value is `"linear"`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 714.

Increases the resolution of the spatial map by factor, changing the dimensions of the spatial map's grid of values (while leaving its spatial bounds unchanged), by interpolating new values between the existing values. The parameter factor must be an integer in [2, 10001], somewhat arbitrarily. The target spatial map is returned, to allow easy chaining of operations. For a 1D spatial map, factor-1 new values will be inserted between every pair of values in the original value grid. A factor of 2 would therefore insert one new value between each pair of existing values, thereby increasing the map's resolution by a factor of two. Note that if the spatial map's original grid dimension was N, the new grid dimension with a factor of k would be k(N−1)+1, not kN, because new values are inserted only between existing values. For 2D and 3D spatial maps, essentially the same process is conducted along each axis of the map's spatiality, increasing the resolution of the map by factor in every dimension. If method is "linear" (the default), linear (or bilinear or trilinear, for 2D/3D maps) interpolation will be used to interpolate the values for the new grid points. Alternatively, if method is "nearest", the nearest value in the old grid will be used for new grid points; with this method, it is recommended that factor be odd, not even, to avoid

artifacts due to rounding of coordinates midway between the original grid positions. If method is "cubic", cubic (or bicubic, for 2D maps) will be used; this generally produces smoother interpolation with fewer artifacts than "linear", but it is not supported for 3D maps. The choice of interpolation method used here is independent of the map's interpolate property. Note that while the "nearest" and "linear" interpolation methods will leave the range of values in the map unchanged, "cubic" interpolation may produce interpolated values that are outside the original range of values (by design). Periodic boundaries are currently supported only for "nearest", "linear", and 1D "cubic" interpolation.

Value

An object of type SpatialMap object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other SpatialMap: SM, add(), blend(), changeColors(), changeValues(), divide(), exp(), gridValues(), mapColor(), mapImage(), mapValue(), multiply(), power(), range(), rescale(), sampleImprovedNearbyPoint(), sampleNearbyPoint(), smooth(), subtract()

IT *InteractionType*

Description

Documentation for InteractionType class from SLiM

Details

This class represents a type of interaction between individuals. This is an advanced feature, the use of which is optional. Once an interaction type is set up with initializeInteractionType() (see section 25.1), it can be evaluated and then queried to give information such as the nearest interacting neighbors of an individual, or the total strength of interactions felt by an individual, relatively efficiently. There are two types of individual, in the paradigm provided by InteractionType: the receiver of an interaction, and the exerter of that interaction. The same individual might be a receiver for one interaction and the exerter for

another interaction, and both of those interactions might be governed by the same InteractionType, but nevertheless, for any given interaction the distinction remains important. The distinction is important because InteractionType enforces this directionality - from exerters, to receivers - throughout its design. Interactions therefore fundamentally define a one-to-many relationship, from one receiver to the (potentially) many exerters that act upon that receiver. Note that a receiver will never be an exerter of an interaction upon itself; a given individual is never both receiver and exerter in the same interaction. When using InteractionType, you will generally start with a receiver, and ask an InteractionType object to handle a query about that receiver, such as "what are the ten nearest exerters to this receiver?". InteractionType is optimized to find and return the set of exerters influencing a given receiver; it is not optimized for the reverse, finding and returning the set of receivers influenced by a given exerter. (If that seems desirable, you might wish to flip your perspective and regard the interaction as actually working in the opposite direction!) Interactions are usually spatial, depending upon the spatial dimensionality established with initializeSLiMOptions() (section 25.1), but they do not have to be spatial. For spatial interactions, the strength of the interaction from an exerter to a receiver often depends (partly, at least) upon the distance d between the two; nearby exerters often wield a stronger influence upon a receiver than more distant exerters do. Non-spatial interactions, on the other hand, are of course unrelated to distance, and the strength of interaction between two individuals depends entirely upon other factors, expressed by an interaction() callback in the model script. Spatial interactions can use interaction() callbacks too, to modify interaction strengths calculated by SLiM, if factors other than distance need to influence the strength of interactions. Note that if there are N receivers to be assessed, each of which potentially interacts with M possible exerters, then - depending upon the queries executed - InteractionType may take computational time proportional to N×log(N)×M. If every individual interacts with every other, M is equal to N and there will be N2 interactions to be evaluated, and the overall computational time may be as bad as N2log(N), although in practice it is perhaps closer to N2 - still bad. Modeling interactions with large population sizes can therefore be very expensive, although InteractionType goes to considerable lengths to minimize the overhead. To reduce this computational burden for your models, it is essential to reduce N, M or both. Spatial interactions can have - and almost always should have - a maximum distance, which allows them to be evaluated much more efficiently (since all interactions beyond the maximum distance can be assumed to have a strength of zero); setting this maximum distance to be as small as possible, without introducing unwanted artifacts, is the single most important factor for using InteractionType efficiently. For sexual models, interactions that are specific to the sexes in a particular way (males competing with other males, males competing to mate with females, etc.) can be declared to be sex-specific, which can also substantially reduce the overhead of querying them. This "sex-segregation" is just one facet of a larger feature, called "interaction constraints", that allows you to specify a variety of constraints upon which individuals can be exerters and which individuals can be receivers; these constraints can include not only sex, but also age, migrant status, and the values of Individual properties such as tag and tagL0 - tagL4. Setting up constraints allows you to model the interaction of just a subset of a subpopulation with a different subset of a (perhaps different) subpopulation. We will focus, in the remaining discussion, on spatial interactions since they are more common. The first step in InteractionType's evaluation of a spatial interaction is to determine the distance from the individual receiving the interaction (the receiver) to the individual exerting the interaction (the exerter). This is computed as the Euclidean distance between the spatial positions of

the individuals, based upon the spatiality of the interaction (i.e., the spatial dimensions used by the interaction, which may be less than the dimensionality of the simulation as a whole). If the receiver and exerter occupy different subpopulations, it is assumed that they nevertheless occupy the same coordinate system; this can be particularly useful for evaluating interactions between individuals of different species. Second, this distance is compared to the maximum distance for the interaction type; if it is beyond that limit, the interaction strength is always zero (and it is also always zero for the interaction of an individual with itself, since a given individual is never both receiver and exerter, as mentioned above). At this stage, the receiver and exerter are then considered "neighbors" - individuals that are within the maximum interaction distance. Some spatial queries stop at this stage, processing the neighbors of receivers. Third, receiver and exerter constraints are applied, potentially disqualifying some interactions; interacting pairs that remain after this cull are called "interacting neighbors". Some queries stop at this stage, processing the interacting neighbors of receivers. Fourth, for all interacting neighbors, the distance is converted into an interaction strength by an interaction function (IF), which is a characteristic of the InteractionType. Finally, this interaction strength may be modified by the interaction() callbacks currently active in the simulation, if any (see section 26.7). Some queries stop at this stage, processing the interaction strengths between interacting neighbors. InteractionType is actually a wrapper for three different spatial query engines that share some of their data but work very differently. The first engine is a brute-force engine that simply computes distances and interaction strengths in response to queries. This engine is usually used in response to queries for simple information, such as the distance(), distanceToPoint(), and strength() methods. The second engine is based upon a data structure called a "k-d tree" that is designed to optimize searches for spatially proximate points. This engine is usually used directly in response to queries involving "neighbors", such as nearestNeighbors() and nearestNeighborsOfPoint(). As mentioned above, the term "neighbor" means an individual that is within the maximum interaction distance of a receiver (excluding the receiver itself) or a focal point; the neighbors of the receiver or point are therefore those that fall within the fixed radius defined by the maximum interaction distance. Calls with "neighbor" in their name explicitly use the k-d tree engine, and may therefore be called only for spatial interactions; in non-spatial interactions there is no concept of a "neighbor". In terms of computational complexity, finding the nearest neighbor of a given receiver by brute force would be an O(N) computation, whereas with the k-d tree it is typically an O(log N) computation - a very important difference, especially for large N. In general, to get the best performance from a spatial model, you should use neighbor-based calls to make minimal queries when possible. If all you really care about is the distance to the nearest neighbor of a receiver, for example, then use nearestNeighbors() to find the neighbor and then call distance() to get the distance to that neighbor, rather than getting the distances to all individuals with distance() and then using min() to select the smallest. The third engine, introduced in SLiM 3.1 and radically modified in SLiM 4, is based upon a data structure called a "sparse array" that is designed to track sparse non-zero values within a dataset that contains mostly zeros. It applies to spatial interactions because most pairs of individuals probably interact with a strength of zero (because typically N » M, because few exerters fall within the maximum interaction radius from a given receiver). In SLiM 4, the full sparse array of interactions is no longer calculated (as it was in SLiM 3); instead, single rows of the sparse array are calculated on demand, providing most of the benefits of the data structure with only a tiny fraction of the memory overhead. In InteractionType parlance, such a single row of the sparse array is called a sparse vector. Sparse vectors are

used to temporarily cache calculated distances and strengths for interactions within a given subpopulation. They are built using the k-d tree to find the interacting neighbors of each individual, and once built they can respond extremely quickly to queries from methods such as totalOfNeighborStrengths(); the interacting neighbors of a given individual are already known, allowing response in O(M) time. These sparse vectors are built on demand, when queries that would benefit from them are made. For them to be effective, it is particularly important that a maximum interaction distance be used that is as small as possible, so beginning with SLiM 3.1 a warning is issued when no maximum distance is defined for spatial interactions. As mentioned above, once an interaction distance has been found it is translated into an interaction strength by an interaction function that depends upon the distance d between individuals. There are currently six options for interaction functions (IFs) in SLiM, represented by single-character codes: "f" - a fixed interaction strength. This IF type has a single parameter, the interaction strength to be used for all interactions of this type. By default, interaction types use a type "f" IF with a value of 1.0, so interactions are binary: on within the maximum distance, off outside. "l" - a linear interaction strength. This IF type has a single parameter, the maximum interaction strength to be used at distance 0.0. The interaction strength falls off linearly, reaching exactly zero at the maximum distance. In other words, for distance d, maximum interaction distance dmax, and maximum interaction strength fmax, the formula for this IF is f(d) = fmax(1 − d / dmax). "e" - A negative exponential interaction strength. This IF type is specified by two parameters, a maximum interaction strength and a rate (inverse scale) parameter. The interaction strength falls off non-linearly from the maximum, and cuts off discontinuously at the maximum distance; typically a maximum distance is chosen such that the interaction strength at that distance is very small anyway. The IF for this type is f(d) = fmaxexp(− d), where is the specified rate parameter. Note that this parameterization is not the same as for the Eidos function rexp(). "n" - A normal interaction strength (i.e., Gaussian, but "g" is avoided to prevent confusion with the gamma-function option provided for, e.g., MutationType). The interaction strength falls off non-linearly from the maximum, and cuts off discontinuously at the maximum distance; typically a maximum distance is chosen such that the interaction strength at that distance is very small anyway. This IF type is specified by two parameters, a maximum interaction strength and a standard deviation. The Gaussian IF for this type is f(d) = fmaxexp(−d2/2 2), where is the standard deviation parameter. Note that this parameterization is not the same as for the Eidos function rnorm(). A Gaussian function is often used to model spatial interactions, but is relatively computation-intensive. "c" - A Cauchy-distributed interaction strength. The interaction strength falls off non-linearly from the maximum, and cuts off discontinuously at the maximum distance; typically a maximum distance is chosen such that the interaction strength at that distance is very small anyway. This IF type is specified by two parameters, a maximum interaction strength and a scale parameter. The IF for this type is f(d) = fmax/(1+(d/)2), where is the scale parameter. Note that this parameterization is not the same as for the Eidos function rcauchy(). A Cauchy distribution can be used to model interactions with relatively fat tails, but for spatiality greater than 1D, the t-distribution (type "t") with degrees of freedom larger than one might be a better choice since the interaction function as we define it with the Cauchy distribution is not normalizable in more than one dimension. "t" - A t-distributed interaction strength. The interaction strength falls off non-linearly from the maximum, and cuts off discontinuously at the maximum distance; typically a maximum distance is chosen such that the interaction strength at that distance is very small anyway. This IF type is specified by three parameters: a maximum interaction strength, the "degrees

of freedom" of the tdistribution (which must be greater than the spatial dimension minus one), and a scale parameter . The IF for this type is f(d) = fmax/(1+(d/)2/)−(+1)/2. The t-distribution can be used to model interactions with relatively fat tails. An InteractionType may be created using the initializeInteractionType() function (see section 25.1). It is often then configured with a specific interaction function, using setInteractionFunction(), and perhaps a set of interaction constraints for receivers and/or exerters, using setConstraints(). It must then be evaluated, with the evaluate() method, for the subpopulations containing exerters and receivers before it will respond to queries regarding individuals in those subpopulations; querying with exerters or receivers whose subpopulations have not been evaluated will result in an error. Calling evaluate() causes the positions of all receivers and exerters to be cached, thus defining a snapshot in time that the InteractionType will then use to respond to queries (allowing it to build its k-d tree based upon the snapshot positions). In WF models, this evaluated state will last until the current parental generation expires, at the end of the next offspring-generation stage. Before the InteractionType may be used with the new parental generation (the offspring of the old parental generation), the interaction must be evaluated again. In nonWF models, interactions are invalidated twice during the tick cycle: once after new offspring are produced (just before early() events), and once just before individuals die (just after fitness calculations); they are also invalidated any time takeMigrants() is called to move individuals between subpopulations. Note that interaction() callbacks are called when queries are served, not when evaluate() is called. As mentioned above, InteractionType supports eligibility constraints for both receivers and exerters, configured with setConstraints(). The snapshot taken by evaluate() also encompasses information about which individuals satisfy any configured exerter constraints; changing the state of individuals after evaluate() is called will not change whether those individuals are qualified to act as exerters in interactions. However - caveat lector - it does not encompass the corresponding constraint information about receivers! Changing the state of individuals (in particular, their tag / tagL0 / tagL1 / tagL2 / tagL3 / tagL4 values) after evaluate() is called may cause individuals to become qualified, or to become disqualified, to act as receivers, because receiver constraints are evaluated at query time, not at evaluate() time. This small discrepancy in InteractionType's conceptual model was chosen for performance reasons, but it can be regarded as a feature, not a bug, since it allows the receivers for each query to be tailored without re-evaluating the interaction type. In any case, if you need receiver constraints to be cached at evaluate() time you can evaluate those constraints yourself, in script, and cache the result in an Individual property such as tagL0, and then use that property as the receiver criterion for the interaction, thereby using the eligibility status that you precalculated and cached. This approach is also useful for enforcing complex eligibility constraints, for either receivers or exerters, that go beyond what InteractionType supports intrinsically; you can always calculate eligibility yourself, cache the result in a property such as tagL0, and use that property as the interaction type's receiver constraint. InteractionType will automatically account for any periodic spatial boundaries established with the periodicity parameter of initializeSLiMOptions(); interactions will wrap around the periodic boundaries without any additional configuration of the interaction. Interactions involving periodic spatial boundaries entail some additional overhead in both memory usage and processor time; in particular, setting up the k-d tree after the interaction is evaluated may take many times longer than in the non-periodic case (but this is rarely a bottleneck anyway since it is quite fast). Once the k-d tree has been set up, responses to spatial queries involving it should then be nearly as fast as in the non-periodic case. Spatial queries that do not involve the k-d tree will generally be marginally

slower than in the non-periodic case, but the difference should not be large. InteractionType provides a fairly large and confusingly similar set of methods, designed to answer every common type of spatial query efficiently. To help find the right method for the job, here is a summary of the methods that involve distances or interaction strengths, either between receivers and exerters, or between a focal point and exerters: Consider whether you want to query for neighbors (all individuals near the receiver), for interacting neighbors (nearby individuals that exert an interaction strength upon the receiver, regardless of what that strength is), or for something about the actual interaction strengths. In general, the simpler queries will be faster; finding neighbors or interacting neighbors is going to be faster than actually evaluating strengths. Furthermore, counting individuals is faster than actually returning the individuals in question. The last three methods in the table have to evaluate the interaction strength between a receiver and every exerter that interacts with it, so they can be fairly slow; if you can, for example, simply count the number of neighbors with neighborCount(), or count the number of interacting neighbors with interactingNeighborCount(), rather than using totalOfNeighborStrengths() to sum up their interaction strengths, the former alternatives will likely be significantly faster than the latter. Finally, as has been mentioned above, for best performance the maximum interaction distance should be set to as small a value as possible; this point is crucial for performance. This class has the following methods (functions):

- `clippedIntegral`
- `distance`
- `distanceFromPoint`
- `drawByStrength`
- `evaluate`
- `interactingNeighborCount`
- `interactionDistance`
- `localPopulationDensity`
- `nearestInteractingNeighbors`
- `nearestNeighbors`
- `nearestNeighborsOfPoint`
- `neighborCount`
- `neighborCountOfPoint`
- `setConstraints`
- `setInteractionFunction`
- `strength`
- `testConstraints`
- `totalOfNeighborStrengths`
- `unevaluate`

This class has the following properties:

id A property of type integer or float or logical or string or string or integer. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The identifier for this interaction type; for interaction type i3, for example, this is 3.

maxDistance A property of type integer or float or logical or string or string or integer. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** The maximum distance over which this interaction will be evaluated. For inter-individual distances greater than maxDistance, the interaction strength will be zero.

reciprocal A property of type integer or float or logical or string or string or integer. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The reciprocality of the interaction, as specified in initializeInteractionType(). This will be T for reciprocal interactions (those for which the interaction strength of B upon A is equal to the interaction strength of A upon B), and F otherwise.

sexSegregation A property of type integer or float or logical or string or string or integer. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The sex-segregation of the interaction, as specified in initializeInteractionType() or with setConstraints(). For non-sexual simulations, this will be "**". For sexual simulations, this string value indicates the sex of individuals feeling the interaction, and the sex of individuals exerting the interaction; see initializeInteractionType() for details.

spatiality A property of type integer or float or logical or string or string or integer. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The spatial dimensions used by the interaction, as specified in initializeInteractionType(). This will be "" (the empty string) for non-spatial interactions, or "x", "y", "z", "xy", "xz", "yz", or "xyz", for interactions using those spatial dimensions respectively. The specified dimensions are used to calculate the distances between individuals for this interaction. The value of this property is always the same as the value given to initializeInteractionType().

tag A property of type integer or float or logical or string or string or integer. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** A user-defined integer value. The value of tag is initially undefined, and it is an error to try to read it; if you wish it to have a defined value, you must arrange that yourself by explicitly setting its value prior to using it elsewhere in your code. The value of tag is not used by SLiM; it is free for you to use. See also the getValue() and setValue() methods (provided by the Dictionary class; see the Eidos manual), for another way of attaching state to interaction types.

See Also

Other InteractionType: `clippedIntegral()`, `distanceFromPoint()`, `distance()`, `drawByStrength()`, `evaluate()`, `interactingNeighborCount()`, `interactionDistance()`, `localPopulationDensity()`, `nearestInteractingNeighbors()`, `nearestNeighborsOfPoint()`, `nearestNeighbors()`, `neighborCountOfPoint()`, `neighborCount()`, `setConstraints()`, `setInteractionFunction()`, `strength()`, `testConstraints()`, `totalOfNeighborStrengths()`, `unevaluate()`

`killIndividuals` *SLiM method killIndividuals*

Description

Documentation for SLiM function `killIndividuals`, which is a method of the SLiM class `Species`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
killIndividuals(individuals)
```

Arguments

`individuals`	An object of type Individual object. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 720.

Immediately kills the individuals in individuals. This removes them from their subpopulation and gives them an index value of -1. The Individual objects are not freed immediately, since references to them could still exist in local Eidos variables; instead, the individuals are kept in a temporary "graveyard" until they can be freed safely. It therefore continues to be safe to use them and their genomes, except that accessing their subpopulation property will raise an error since they no longer have a subpopulation. Note that the indices and order of individuals and genomes in all source subpopulations will change unpredictably as a side effect of this method. All evaluated interactions are invalidated as a side effect of calling this method. Note that this method is only for use in nonWF models, in which mortality is managed manually by the model script. In WF models, mortality is managed automatically by the SLiM core when the new offspring generation becomes the parental generation and the previous parental generation dies; mortality does not otherwise occur in WF models. In nonWF models, mortality normally occurs during the survival stage of the tick cycle (see section 24.4), based upon the fitness values calculated by SLiM, and survival() callbacks can influence the outcome of that survival stage. Calls to killIndividuals(), on the other hand, can be made at any time during first(), early(), or late() events, and the result cannot be modified by survival() callbacks; the given individuals are simply immediately killed. This method therefore provides an alternative, and relatively rarely used, mortality mechanism that is disconnected from fitness.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Species: Sp, addSubpopSplit(), addSubpop(), countOfMutationsOfType(), individualsWithPedigreeID
mutationCounts(), mutationFrequencies(), mutationsOfType(), outputFixedMutations(),
outputFull(), outputMutations(), readFromPopulationFile(), recalculateFitness(),
registerFitnessEffectCallback(), registerMateChoiceCallback(), registerModifyChildCallback(),
registerMutationCallback(), registerMutationEffectCallback(), registerRecombinationCallback(),
registerReproductionCallback(), registerSurvivalCallback(), simulationFinished(),
skipTick(), subsetMutations(), treeSeqCoalesced(), treeSeqOutput(), treeSeqRememberIndividuals(),
treeSeqSimplify()

---

late *SLiM late() callback*

Description

This callback specifies the code should be called late in the simulation cycle. For details on exactly when first(), late() and early() callbacks are run during a simulation see SLiM Manual: page 541 for "WF" models, or SLiM Manual: page 549 for "nonWF" models.

Usage

```
late()
```

Value

None

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other callbacks: `early()`, `first()`, `fitnessEffect()`, `fitness()`, `initialize()`, `interaction()`, `mateChoice()`, `modifyChild()`, `mutationEffect()`, `mutation()`, `recombination()`, `reproduction()`, `slim_callbacks()`, `survival()`

Examples

```
slim_block(100, late(), {
  sim.simulationFinished()
})
```

LF *LogFile*

Description

Documentation for LogFile class from SLiM

Details

LogFile is a class that can, optionally, be used to log out a table of information about the running simulation to a text file. The information logged out is completely configurable with generators, including the ability to use custom Eidos code as a generator. The resulting file can be formatted with comma separators (a CSV file), tab separators (a TSV file), or with any custom separator string between data values. The file can be plain text or can be compressed in .gz format (decompressible at the command line with the gunzip utility, among other tools). When compression is enabled, writes out to disk are buffered in memory for better performance and smaller file size. A new LogFile object attached to the simulation can be created with the Community method createLogFile(), discussed in section 25.3.2. Any number of LogFile objects can be active simultaneously, writing to different files. Logging can be done automatically, at the end of ticks at a given periodic interval (e.g., every 10th tick). This automatic logging is optional, and a log row can always be generated explicitly by called logRow(). Flushing compressed data to disk can be done automatically after a given number of rows have been generated, or can be requested explicitly with a call to flush(). Generators always generate a single value in each logged row, resulting in a single column of data in the log file. Some built-in generators are provided by LogFile for the most common cases; these can be added to a given LogFile by calling the add...() methods documented below. A generator with custom Eidos code can be added with addCustomColumn(), and a generator that expects to be supplied with the value to write, rather than generating the value itself, can be added with addSuppliedColumn(). LogFile expects to be fully configured, with calls to add...() methods to add generators, before the first row of data is written out, to ensure consistency in the file's data. When the first row of data is written (or buffered), the LogFile's configuration will then be frozen, and calls

to add...() will no longer be allowed. Columns will be written, in each row, in the order in which add...() calls are made on the LogFile, and they will be named in the file's header line as documented in those methods. It is an error to have two columns with the same name. LogFile is a subclass of the Eidos class Dictionary, but unlike other SLiM classes that are Dictionary subclasses, this does not allow you to attach arbitrary key-value pairs to the object. Instead, LogFile uses its Dictionary-ness to make the data from the last logged row available through getValue(), using the name of the generator (i.e., the name of the data column) as the key. One quirk of LogFile is that because its keys are defined by the columns that it generates, and columns can sometimes contain an NA value, LogFile's dictionary can, in effect, contain NULL values (representing NA); this is not normally allowed by Dictionary. This should cause no trouble; just be aware that getValue() on a LogFile might return NULL for a key, representing NA, and that the result of serialize() might contain NA values. Finally: as a general rule, if a Subpopulation is referenced by one of the add...X() methods below it may be supplied as an object if it already exists (p1), or by id (1) even if it does not yet exist. The resulting column will generally have a name of the form pX_colname, where X is the id of the specified subpopulation. If a subpopulation-specific data logger refers to a Subpopulation that does not exist at the time a given row is logged, NA will be written. This class has the following methods (functions):

- `addCustomColumn`
- `addCycle`
- `addCycleStage`
- `addKeysAndValuesFrom`
- `addMeanSDColumns`
- `addPopulationSexRatio`
- `addPopulationSize`
- `addSubpopulationSexRatio`
- `addSubpopulationSize`
- `addSuppliedColumn`
- `addTick`
- `clearKeysAndValues`
- `flush`
- `logRow`
- `setLogInterval`
- `setFilePath`
- `setSuppliedValue`
- `setValue`
- `willAutolog`

This class has the following properties:

allKeys A property of type string or string or integer or integer or integer. This property is a constant, so it is not modifiable. **Property Description:** This Dictionary property has an override in LogFile to return, in order, the names of the log file columns, which are the keys for LogFile's dictionary.

filePath A property of type string or string or integer or integer or integer. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The path of the log file being written to. This may be changed with setFilePath().

logInterval A property of type string or string or integer or integer or integer. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The interval for automatic logging; a new row of data will be logged every logInterval ticks. This may be changed with setLogInterval(). If automatic logging has been disabled, this property will be 0.

precision A property of type string or string or integer or integer or integer. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** The precision of float output. To be exact, precision specifies the preferred number of significant digits that will be output for float values. The default is 6; values in [1,22] are legal, but 17 is probably the largest value that makes sense given the limits of double-precision floating point.

tag A property of type string or string or integer or integer or integer. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** A user-defined integer value. The value of tag is initially undefined, and it is an error to try to read it; if you wish it to have a defined value, you must arrange that yourself by explicitly setting its value prior to using it elsewhere in your code. The value of tag is not used by SLiM; it is free for you to use.

See Also

Other LogFile: `addCustomColumn()`, `addCycleStage()`, `addCycle()`, `addKeysAndValuesFrom()`, `addMeanSDColumns()`, `addPopulationSexRatio()`, `addPopulationSize()`, `addSubpopulationSexRatio()`, `addSubpopulationSize()`, `addSuppliedColumn()`, `addTick()`, `clearKeysAndValues()`, `flush()`, `logRow()`, `setFilePath()`, `setLogInterval()`, `setSuppliedValue()`, `setValue()`, `willAutolog()`

`localPopulationDensity`

SLiM method localPopulationDensity

Description

Documentation for SLiM function `localPopulationDensity`, which is a method of the SLiM class `InteractionType`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
localPopulationDensity(receivers, exerterSubpop)
```

Arguments

`receivers`	An object of type Individual object. See details for description.
`exerterSubpop`	An object of type null or Subpopulation object. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 694.

Returns a vector of the local population density present at the location of each individual in receivers, which does not need to be a singleton; indeed, it can be a vector of all of the individuals in a given subpopulation. However, all receivers must be in the same subpopulation. The local population density is computed from exerters in exerterSubpop, or if that is NULL (the default), then from the receiver's subpopulation. The evaluate() method must have been previously called for the receiver and exerter subpopulations, and positions saved at evaluation time will be used. Population density is estimated using interaction strengths, effectively doing a kernel density estimate using the interaction function as the kernel. What is returned is computed as the total interaction strength present at a given point, divided by the integral of the interaction function around that point after clipping by the spatial bounds of the exerter subpopulation (what one might think of as the amount of "interaction field" around the point). This provides an estimate of local population density, in units of individuals per unit area, as a weighted average over the area covered by the interaction function, where the weight of each exerter in the average is the value of the interaction function at that exerter's position. This can also be thought of as a measure of the amount of interaction happening per unit of interaction field in the space surrounding the point. To calculate the clipped integral of the interaction function, this method uses the same numerical estimator used by the clippedIntegral() method of InteractionType, and all of the caveats described for that method apply here also; notably, all individuals must be within spatial bounds, the maximum interaction distance must be less than half the spatial extent of the subpopulation, and interaction() callbacks are not used (and so, for this method, are not allowed to be active). See the documentation for clippedIntegral() for further discussion of the details of these calculations. To calculate the total interaction strength around the position of a receiver, this method uses the same machinery as the totalOfNeighborStrengths() method of InteractionType, except that - in contrast to other InteractionType methods - the interaction strength exerted by the receiver itself is included in the total (if the exerter subpopulation is the receiver's own subpopulation). This is because population density at the location of an individual includes the individual itself. If this is not desirable, the totalOfNeighborStrengths() method should probably be used. To see the point of this method, consider a receiver located near the edge of the spatial bounds of its subpopulation. Some portion of the interaction function that surrounds that receiver falls outside the spatial bounds of its subpopulation, and will therefore never contain an interacting exerter. If, for example, interaction strengths are used as a measure of competition, this receiver will therefore have an advantage, because it will never feel any competition from the portion of its range that falls outside spatial bounds. However, that portion of its range is presumably also not available to the receiver itself, for foraging or hunting, in which case this advantage is not biologically realistic, but is instead just an undesirable "edge effect" artifact. Dividing by the integral of the interaction function, clipped to the spatial bounds, provides a way to compensate for this edge effect. A nice side effect of using local population densities instead of total interaction strengths is that

the maximum interaction strength passed to setInteractionFunction() no longer matters; it cancels out when the total interaction strength is divided by the receiver's clipped integral. However, the shape of the interaction function does still matter; it determines the relative weights used for exerters at different distances from the position of the receiver.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other InteractionType: IT, clippedIntegral(), distanceFromPoint(), distance(), drawByStrength(), evaluate(), interactingNeighborCount(), interactionDistance(), nearestInteractingNeighbors(), nearestNeighborsOfPoint(), nearestNeighbors(), neighborCountOfPoint(), neighborCount(), setConstraints(), setInteractionFunction(), strength(), testConstraints(), totalOfNeighborStrengths unevaluate()

---

logRow *SLiM method logRow*

Description

Documentation for SLiM function logRow, which is a method of the SLiM class LogFile. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
logRow(void)
```

Arguments

void	An object of type . See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 703.

This logs a new row of data, by evaluating all of the generators added to the LogFile with add...() calls. Note that the new row may be buffered, and thus may not be written out to disk immediately; see flush(). This method may be used instead of, or in conjunction with, automatic logging. You can get the LogFile instance, in order to call logRow() on it, from community.logFiles, or you can remember it in a global constant with defineConstant().

Value

An object of type void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or logical.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other LogFile: LF, addCustomColumn(), addCycleStage(), addCycle(), addKeysAndValuesFrom(), addMeanSDColumns(), addPopulationSexRatio(), addPopulationSize(), addSubpopulationSexRatio(), addSubpopulationSize(), addSuppliedColumn(), addTick(), clearKeysAndValues(), flush(), setFilePath(), setLogInterval(), setSuppliedValue(), setValue(), willAutolog()

M *Mutation*

Description

Documentation for Mutation class from SLiM

Details

This class represents a single point mutation. Mutations can be shared by the genomes of many individuals; if they reach fixation, they are converted to Substitution objects. Although Mutation has a tag property, like most SLiM classes, the subpopID can also store custom values if you don't need to track the origin subpopulation of mutations (see below). Section 1.5.2 presents an overview of the conceptual role of this class. This class has the following methods (functions):

- setMutationType
- setSelectionCoeff

This class has the following properties:

id A property of type integer. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The identifier for this mutation. Each mutation created during a run receives an immutable identifier that will be unique across the duration of the run. These identifiers are not re-used during a run, except that if a population file is loaded from disk, the loaded mutations will receive their original identifier values as saved in the population file.

isFixed A property of type logical. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** T if the mutation has fixed (in the SLiM sense of having been converted to a Substitution object), F otherwise. Since fixed/substituted mutations are removed from the simulation, you will only see this flag be T if you have held onto a mutation beyond its usual lifetime (see section 28.2).

isSegregating A property of type logical. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** T if the mutation is segregating (in the SLiM sense of not having been either lost or converted to a Substitution object), F otherwise. Since both lost and fixed/substituted mutations are removed from the simulation, you will only see this flag be F if you have held onto a mutation beyond its usual lifetime (see section 28.2). Note that if isSegregating is F, isFixed will let you determine whether the mutation is no longer segregating because it was lost, or because it fixed.

mutationType A property of type MutationType object. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The MutationType from which this mutation was drawn.

nucleotide A property of type string. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** A string representing the nucleotide associated with this mutation; this will be "A", "C", "G", or "T". If the mutation is not nucleotide-based, this property is unavailable.

nucleotideValue A property of type integer. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** An integer representing the nucleotide associated with this mutation; this will be 0 (A), 1 (C), 2 (G), or 3 (T). If the mutation is not nucleotide-based, this property is unavailable.

originTick A property of type integer. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The tick in which this mutation arose.

position A property of type integer. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The position in the chromosome of this mutation.

selectionCoeff A property of type float. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The selection coefficient of the mutation, drawn from the distribution of fitness effects of its MutationType. If a mutation has a selectionCoeff of s, the multiplicative fitness effect of the mutation in a

homozygote is 1+s; in a heterozygote it is 1+hs, where h is the dominance coefficient kept by the mutation type (see section 25.11.1). Note that this property has a quirk: it is stored internally in SLiM using a single-precision float, not the double-precision float type normally used by Eidos. This means that if you set a mutation mut's selection coefficient to some number x, mut.selectionCoeff==x may be F due to floating-point rounding error. Comparisons of floating-point numbers for exact equality is often a bad idea, but this is one case where it may fail unexpectedly. Instead, it is recommended to use the id or tag properties to identify particular mutations.

subpopID A property of type integer. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** The identifier of the subpopulation in which this mutation arose. This property can be used to track the ancestry of mutations through their subpopulation of origin. For an overview of other ways of tracking genetic ancestry, including true local ancestry at each position on the chromosome, see sections 1.7 and 14.7. If you don't care which subpopulation a mutation originated in, the subpopID may be used as an arbitrary integer "tag" value for any purpose you wish; SLiM does not do anything with the value of subpopID except propagate it to Substitution objects and report it in output. (It must still be >= 0, however, since SLiM object identifiers are limited to nonnegative integers).

tag A property of type integer. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** A user-defined integer value. The value of tag is initially undefined, and it is an error to try to read it; if you wish it to have a defined value, you must arrange that yourself by explicitly setting its value prior to using it elsewhere in your code. The value of tag is not used by SLiM; it is free for you to use.

See Also

Other Mutation: `setMutationType()`, `setSelectionCoeff()`

`mapColor` *SLiM method mapColor*

Description

Documentation for SLiM function `mapColor`, which is a method of the SLiM class `SpatialMap`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
mapColor(value)
```

Arguments

`value`	An object of type numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 714.

Uses the spatial map's color-translation machinery (as defined by the valueRange and colors parameters to defineSpatialMap()) to translate each element of value into a corresponding color string. If the spatial map does not have color-translation capabilities, an error will result. See the documentation for defineSpatialMap() for information regarding the details of color translation. See the Eidos manual for further information on color strings.

Value

An object of type string.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other SpatialMap: SM, add(), blend(), changeColors(), changeValues(), divide(), exp(), gridValues(), interpolate(), mapImage(), mapValue(), multiply(), power(), range(), rescale(), sampleImprovedNearbyPoint(), sampleNearbyPoint(), smooth(), subtract()

---

mapImage *SLiM method mapImage*

Description

Documentation for SLiM function mapImage, which is a method of the SLiM class SpatialMap. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
mapImage(width, height, centers, color)
```

Arguments

`width`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`height`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`centers`	An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.
`color`	An object of type logical. Must be of length 1 (a singleton). The default value is `T`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 715.

Returns an Image object sampled from the spatial map. The image will be width pixels wide and height pixels tall; the intrinsic size of the spatial map itself will be used if one of these parameters is NULL. The image will be oriented in the same way as it is displayed in SLiMgui (which conceptually entails a transformation from matrix coordinates, which store values by column, to standard image coordinates, which store values by row; see the Eidos manual's documentation of Image for details). This method may only be called for 2D spatial maps at present. The sampling of the spatial map can be done in one of two ways, as controlled by the centers parameter. If centers is T, a (width+1) × (height+1) grid of lines that delineates width × height rectangular pixels will be overlaid on top of the spatial map, and values will be sampled from the spatial map at the center of each of these pixels. If centers is F (the default), a width × height grid of lines will be overlaid on top of the spatial map, and values will be sampled from the spatial map at the vertices of the grid. If interpolation is not enabled for the spatial map, these two options will both recover the original matrix of values used to define the spatial map (assuming, here and below, that width and height are NULL). If interpolation is enabled for the spatial map, however, centers == F will recover the original values, but will not capture the "typical" value of each pixel in the image; centers == T, on the other hand, will not recover the original values, but will capture the "typical" value of each pixel in the image (i.e., the value at the center of each pixel, as produced by interpolation). The figures in section 16.11 may be helpful for visualizing the difference between these options; the overlaid grids span the full extent of the spatial map, just as shown in that section. If color is T (the default), the valueRange and colors parameters supplied to defineSpatialMap() will be used to translate map values to RGB color values as described in the documentation of that method, providing the same appearance as in SLiMgui; of course those parameters must have been supplied, otherwise an error will result. If color is F, on the other hand, a grayscale image will be produced that directly reflects the map values without color translation. In this case, this method needs to translate map values, which can have any float value, into grayscale pixel values that are integers in [0, 255]. To do so, the map values are multiplied by 255.0, clamped to [0.0, 255.0], and then rounded to the nearest integer. This translation scheme essentially assumes that map values are in [0, 1]; for spatial maps that were defined using the floatK channel of a grayscale PNG image, this should recover the original image's pixel values. (If a different translation scheme is desired, color=T with the desired valueRange and colors should be used.)

Value

An object of type Image object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other SpatialMap: SM, add(), blend(), changeColors(), changeValues(), divide(), exp(), gridValues(), interpolate(), mapColor(), mapValue(), multiply(), power(), range(), rescale(), sampleImprovedNearbyPoint(), sampleNearbyPoint(), smooth(), subtract()

---

mapValue *SLiM method mapValue*

Description

Documentation for SLiM function `mapValue`, which is a method of the SLiM class SpatialMap. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
mapValue(point)
```

Arguments

`point`	An object of type float. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 715.

Uses the spatial map's mapping machinery (as defined by the gridSize, values, and interpolate parameters to defineSpatialMap()) to translate the coordinates of point into a corresponding map value. The length of point must be equal to the spatiality of the spatial map; in other words, for a spatial map with spatiality "xz", point must be of length 2,

specifying the x and z coordinates of the point to be evaluated. Interpolation will automatically be used if it was enabled for the spatial map. Point coordinates are clamped into the range defined by the spatial boundaries, even if the spatial boundaries are periodic; use pointPeriodic() to wrap the point coordinates first if desired. See the documentation for defineSpatialMap() for information regarding the details of value mapping. Beginning in SLiM 3.3, point may contain more than one point to be looked up. In this case, the length of point must be an exact multiple of the spatiality of the spatial map; for a spatial map with spatiality "xz", for example, the length of point must be an exact multiple of 2, and successive pairs of elements from point (elements 0 and 1, then elements 2 and 3, etc.) will be taken as the x and z coordinates of the points to be evaluated. This allows spatialMapValue() to be used in a vectorized fashion. The spatialMapValue() method of Subpopulation provides the same functionality as this method; it may be more convenient to use, for some usage cases, and it checks that the spatial map is actually added to the subpopulation in question, providing an additional consistency check. However, either method may be used.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other SpatialMap: SM, add(), blend(), changeColors(), changeValues(), divide(), exp(), gridValues(), interpolate(), mapColor(), mapImage(), multiply(), power(), range(), rescale(), sampleImprovedNearbyPoint(), sampleNearbyPoint(), smooth(), subtract()

---

mateChoice *SLiM mateChoice() callback*

Description

This callback specifies that a code block is providing logic for an individual to choose a mate. The first parent is chosen based on fitness values, then this callback is called to determine the second parent (if no callback is specified, it is chosen randomly proportional to fitness). The callback code should return a vector of weights specifying the proportional

probability of each individual in the population being the second parent (including the first parent, as selfing is possible). Alternatively, it could return an `Individual` object referring to the individual chosen as the second parent. Returning a numeric `0` tells SLiM that no mate could be found. Lastly, returning `NULL` tells SLiM to use the default `weights` vector, which is the same as just returning `weights` but is more efficient. see SLiM Manual: page 599

Usage

```
mateChoice(subpop_id)
```

Arguments

`subpop_id`	The id(s) of the subpopulation(s) to which this callback should apply. Can be an integer 1, 2, etc., or character "p1", "p2", etc.

Details

Global variables available in reproduction callbacks:

individual The parent already chosen (the female, in sexual simulations)

genome1 One genome of the parent already chosen

genome2 The other genome of the parent already chosen

subpop The subpopulation into which the offspring will be placed

sourceSubpop The subpopulation from which the parents are being chosen

weights The standard fitness-based weights for all individuals

Value

None

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other callbacks: `early()`, `first()`, `fitnessEffect()`, `fitness()`, `initialize()`, `interaction()`, `late()`, `modifyChild()`, `mutationEffect()`, `mutation()`, `recombination()`, `reproduction()`, `slim_callbacks()`, `survival()`

Examples

```
slim_block(mateChoice(p2), {

  ## in subpopulation 2, high fitness individuals are preferred
  return(weights^2)

})
```

---

minimal_slimr_script *Generate a minimal slimr_script object*

Description

This function is mainly used to quickly create a script which does not does not do much, but can be used to test other code that work on 'slimr_script' objects. Mainly used for documentation examples.

Usage

```
minimal_slimr_script(command = print("Hello world!"))
```

Arguments

```
command
```

Value

A 'slimr_script' object

Examples

```
minimal_slimr_script()
```

---

minimal_slim_sim *Generate a minimal SLiM simulation script*

Description

This function is mainly used to quickly create a script which does not does not do much, but can be used to test other code.

Usage

```
minimal_slim_sim(command = print("Hello world!"))
```

Arguments

`command`

Value

An R 'expression'.

Examples

```
minimal_slim_sim()
```

`mm16To256` *SLiM method mm16To256*

Description

Documentation for SLiM function `mm16To256`, which is a method of the SLiM class `SLiMBuiltin`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
mm16To256(mutationMatrix16)
```

Arguments

`mutationMatrix16`
: An object of type float. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 749.

Returns a 64×4 mutation matrix that is functionally identical to the supplied 4×4 mutation matrix in mutationMatrix16. The mutation rate for each of the 64 trinucleotides will depend only upon the central nucleotide of the trinucleotide, and will be taken from the corresponding entry for the same nucleotide in mutationMatrix16. This function can be used to easily construct a simple trinucleotide-based mutation matrix which can then be modified so that specific trinucleotides sustain a mutation rate that does not depend only upon their central nucleotide. See the documentation for initializeGenomicElementType() in section 25.1 for further discussion of how these 64×4 mutation matrices are interpreted and used.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other SLiMBuiltin: SB, calcFST(), calcHeterozygosity(), calcInbreedingLoad(), calcPairHeterozygosity(
calcVA(), calcWattersonsTheta(), codonsToAminoAcids(), mmJukesCantor(), mmKimura(),
nucleotideCounts(), nucleotideFrequencies(), nucleotidesToCodons(), summarizeIndividuals(),
treeSeqMetadata()

Examples

```
## This just brings up the documentation:
mm16To256()
```

---

mmJukesCantor *SLiM method mmJukesCantor*

Description

Documentation for SLiM function `mmJukesCantor`, which is a method of the SLiM class SLiMBuiltin. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
mmJukesCantor(alpha)
```

Arguments

alpha	An object of type float. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 749.

Returns a mutation matrix representing a Jukes-Cantor (1969) model with mutation rate alpha to each possible alternative nucleotide at a site: This 2×2 matrix is suitable for use with initializeGenomicElementType(). Note that the actual mutation rate produced by this matrix is 3*alpha.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other SLiMBuiltin: SB, calcFST(), calcHeterozygosity(), calcInbreedingLoad(), calcPairHeterozygosity(
calcVA(), calcWattersonsTheta(), codonsToAminoAcids(), mm16To256(), mmKimura(),
nucleotideCounts(), nucleotideFrequencies(), nucleotidesToCodons(), summarizeIndividuals(),
treeSeqMetadata()

Examples

```
## This just brings up the documentation:
mmJukesCantor()
```

---

mmKimura *SLiM method mmKimura*

Description

Documentation for SLiM function `mmKimura`, which is a method of the SLiM class `SLiMBuiltin`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
mmKimura(alpha, beta)
```

Arguments

`alpha`	An object of type float or float. Must be of length 1 (a singleton). See details for description.
`beta`	An object of type float or float. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 749.

Returns a mutation matrix representing a Kimura (1980) model with transition rate alpha and transversion rate beta: This 2×2 matrix is suitable for use with initializeGenomicElementType(). Note that the actual mutation rate produced by this model is alpha+2*beta.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other SLiMBuiltin: SB, calcFST(), calcHeterozygosity(), calcInbreedingLoad(), calcPairHeterozygosity(
calcVA(), calcWattersonsTheta(), codonsToAminoAcids(), mm16To256(), mmJukesCantor(),
nucleotideCounts(), nucleotideFrequencies(), nucleotidesToCodons(), summarizeIndividuals(),
treeSeqMetadata()

Examples

```
## This just brings up the documentation:
mmKimura()
```

---

modifyChild *SLiM modifyChild() callback*

Description

This callback specifies that a code block is providing logic to modify an offspring that has been produced during a SLiM simulation, and is called for every offspring produced. The code should modify the pseudo-variables associated with the child (see details) `child`, `childGenome1`, and `childGenome2`. It should also return a single logical `T` or `F`. If `F`, the offspring will be discarded and the callback called again. Make sure there is a non-zero probability of returning `T`, or the simulation could hang indefinitely. see SLiM Manual: page 601

Usage

```
modifyChild(subpop_id)
```

Arguments

`subpop_id`	The id(s) of the subpopulation(s) to which this callback should apply. Can be an integer 1, 2, etc., or character "p1", "p2", etc.

Details

Global variables available in reproduction callbacks:

child The generated child (an object of class Individual)

childGenome1 One genome of the generated child

childGenome2 The other genome of the generated child

childIsFemale T if the child will be female, F if male (defined only if sex is enabled)

parent1 The first parent (an object of class Individual)

parent1Genome1 One genome of the first parent

parent1Genome2 The other genome of the first parent

isCloning T if the child is the result of cloning

isSelfing T if the child is the result of selfing

parent2 The second parent (an object of class Individual)

parent2Genome1 One genome of the second parent

parent2Genome2 The other genome of the second parent

subpop The subpopulation in which the child will live

sourceSubpop The subpopulation of the parents (==subpop if not a migration mating)

Value

None

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other callbacks: early(), first(), fitnessEffect(), fitness(), initialize(), interaction(), late(), mateChoice(), mutationEffect(), mutation(), recombination(), reproduction(), slim_callbacks(), survival()

Examples

```
slim_block(modifyChild(), {

  # prevent hermaphroditic selfing
  if(parent1 == parent2) {
    return(F)
  }
  return(T)

})
```

MT *MutationType*

Description

Documentation for MutationType class from SLiM

Details

This class represents a type of mutation with a particular distribution of fitness effects, such as neutral mutations or weakly beneficial mutations. Sections 1.5.3 and 1.5.4 present an overview of the conceptual role of this class. The mutation types currently defined in the simulation are defined as global constants with the same names used in the SLiM input file - m1, m2, and so forth. There are currently six options for the distribution of fitness effects in SLiM, represented by single-character codes: "f" - A fixed fitness effect. This DFE type has a single parameter, the selection coefficient s to be used by all mutations of the mutation type. "g" - A gamma-distributed fitness effect. This DFE type is specified by two parameters, a mean value and a shape parameter. The gamma distribution from which mutations are drawn is given by the probability density function P(s | ,) = [Γ( )]−1s −1exp(−s/), where is the shape parameter, and the specified mean for the distribution is equal to . Note that this parameterization is the same as for the Eidos function rgamma(). A gamma distribution is often used to model deleterious mutations at functional sites. "e" - An exponentially-distributed fitness effect. This DFE type is specified by a single parameter, the mean of the distribution. The exponential distribution from which mutations are drawn is given by the probability density function P(s |) = −1exp(−s/), where is the specified mean for the distribution. This parameterization is the same as for the Eidos function rexp(). An exponential distribution is often used to model beneficial mutations. "n" - A normally-distributed fitness effect. This DFE type is specified by two parameters, a mean and a standard deviation. The normal distribution from which mutations are drawn is given by the probability density function P(s | ,) = (2 2)−1/2exp(−(s−)2/2 2), where is the mean and is the standard deviation. This parameterization is the same as for

the Eidos function rnorm(). A normal distribution is often used to model mutations that can be either beneficial or deleterious, since both tails of the distribution are unbounded. "p" - A Laplace-distributed fitness effect. This DFE type is specified by two parameters, a mean and a scale. The Laplace distribution from which mutations are drawn is given by the probability density function P(s | ,b) = exp(−|s− |/b)/2b, where is the mean and b is the scale parameter. A Laplace distribution is sometimes used to model a mix of both deleterious and beneficial mutations. "w" - A Weibull-distributed fitness effect. This DFE type is specified by a scale parameter and a shape parameter. The Weibull distribution from which mutations are drawn is given by the probability density function P(s | ,k) = (k/ k)sk−1exp(−(s/)k), where is the scale parameter and k is the shape parameter. This parameterization is the same as for the Eidos function rweibull(). A Weibull distribution is often used to model mutations following extreme-value theory. "s" - A script-based fitness effect. This DFE type is specified by a script parameter of type string, specifying an Eidos script to be executed to produce each new selection coefficient. For example, the script "return rbinom(1);" could be used to generate selection coefficients drawn from a binomial distribution, using the Eidos function rbinom(), even though that mutational distribution is not supported by SLiM directly. The script must return a singleton float or integer. Note that these distributions can in principle produce selection coefficients smaller than -1.0. In that case, the mutations will be evaluated as "lethal" by SLiM, and the relative fitness of the individual will be set to 0.0. This class has the following methods (functions):

- `drawSelectionCoefficient`
- `setDistribution`

This class has the following properties:

color A property of type string. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** The color used to display mutations of this type in SLiMgui. Outside of SLiMgui, this property still exists, but is not used by SLiM. Colors may be specified by name, or with hexadecimal RGB values of the form "#RRGGBB" (see the Eidos manual). If color is the empty string, "", SLiMgui's default (selection-coefficient-based) color scheme is used; this is the default for new MutationType objects.

colorSubstitution A property of type string. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** The color used to display substitutions of this type in SLiMgui (see the discussion for the colorSubstitution property of the Chromosome class for details). Outside of SLiMgui, this property still exists, but is not used by SLiM. Colors may be specified by name, or with hexadecimal RGB values of the form "#RRGGBB" (see the Eidos manual). If colorSubstitution is the empty string, "", SLiMgui's default (selection-coefficient-based) color scheme is used; this is the default for new MutationType objects.

convertToSubstitution A property of type logical. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** This property governs whether mutations of this mutation type will be converted to Substitution objects when they reach fixation. In WF models this property is T by default, since conversion to Substitution objects provides large speed benefits; it should be set to F only if necessary, and only on the mutation types for which it is necessary. This might be needed, for example, if you are using a mutationEffect() callback to implement an epistatic relationship between mutations; a mutation epistatically influencing the fitness of other

mutations through a mutationEffect() callback would need to continue having that influence even after reaching fixation, but if the simulation were to replace the fixed mutation with a Substitution object the mutation would no longer be considered in fitness calculations (unless the callback explicitly consulted the list of Substitution objects kept by the simulation). Other scriptdefined behaviors in mutationEffect(), interaction(), mateChoice(), modifyChild(), and recombination() callbacks might also necessitate the disabling of substitution for a given mutation type; this is an important consideration to keep in mind. See section 23.3 for further discussion of convertToSubstitution in WF models. In contrast, for nonWF models this property is F by default, because even mutations with no epistatis or other indirect fitness effects will continue to influence the survival probabilities of individuals. For nonWF models, only neutral mutation types with no epistasis or other side effects can safely be converted to substitutions upon fixation. When such a pure-neutral mutation type is defined in a nonWF model, this property should be set to T to tell SLiM that substitution is allowed; this may have very large positive effects on performance, so it is important to remember when modeling background neutral mutations. See section 24.5 for further discussion of convertToSubstitution in nonWF models. SLiM consults this flag at the end of each tick when deciding whether to substitute each fixed mutation. If this flag is T, all eligible fixed mutations will be converted at the end of the current tick, even if they were previously left unconverted because of the previous value of the flag. Setting this flag to F will prevent future substitutions, but will not cause any existing Substitution objects to be converted back into Mutation objects.

distributionParams A property of type float. This property is a constant, so it is not modifiable. **Property Description:** The parameters that configure the chosen distribution of fitness effects. This will be of type string for DFE type "s", and type float for all other DFE types.

distributionType A property of type string. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The type of distribution of fitness effects; one of "f", "g", "e", "n", "p", "w", or "s" (see section 25.11, above).

dominanceCoeff A property of type float. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** The dominance coefficient used for mutations of this type when heterozygous. Changing this will normally affect the fitness values calculated toward the end of the current tick; if you want current fitness values to be affected, you can call the Species method recalculateFitness() - but see the documentation of that method for caveats. Note that the dominance coefficient is not bounded. A dominance coefficient greater than 1.0 may be used to achieve an overdominance effect. By making the selection coefficient very small and the dominance coefficient very large, an overdominance scenario in which both homozygotes have the same fitness may be approximated, to a nearly arbitrary degree of precision. Note that this property has a quirk: it is stored internally in SLiM using a single-precision float, not the double-precision float type normally used by Eidos. This means that if you set a mutation type muttype's dominance coefficient to some number x, muttype.dominanceCoeff==x may be F due to floating-point rounding error. Comparisons of floating-point numbers for exact equality is often a bad idea, but this is one case where it may fail unexpectedly. Instead, it is recommended to use the id or tag properties to identify particular mutation types.

haploidDominanceCoeff A property of type float. It is of length one (a singleton). This

property is a variable, so it is modifiable. **Property Description:** The dominance coefficient used for mutations of this type when they occur opposite a null genome (as in sex-chromosome models and models involving haploids). This defaults to 1.0, and is used only in models where null genomes are present; the dominanceCoeff property is the dominance coefficient used in most circumstances. Changing this will normally affect the fitness values calculated toward the end of the current tick; if you want current fitness values to be affected, you can call the Species method recalculateFitness() - but see the documentation of that method for caveats. As with the dominanceCoeff property, this is stored internally using a single-precision float; see the documentation for dominanceCoeff for discussion.

id A property of type integer. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The identifier for this mutation type; for mutation type m3, for example, this is 3.

mutationStackGroup A property of type integer. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** The group into which this mutation type belongs for purposes of mutation stacking policy. This is equal to the mutation type's id by default. See mutationStackPolicy, below, for discussion. In nucleotide-based models, the stacking group for nucleotide-based mutation types is always -1, and cannot be changed. Non-nucleotide-based mutation types may also be set to share the -1 stacking group, if they should participate in the same stacking policy as nucleotide-based mutations, but that would be quite unusual.

mutationStackPolicy A property of type string. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** This property and the mutationStackGroup property together govern whether mutations of this mutation type's stacking group can "stack" - can occupy the same position in a single individual. A set of mutation types with the same value for mutationStackGroup is called a "stacking group", and all mutation types in a given stacking group must have the same mutationStackPolicy value, which defines the stacking behavior of all mutations of the mutation types in the stacking group. In other words, one stacking group might allow its mutations to stack, while another stacking group might not, but the policy within each stacking group must be unambiguous. This property is "s" by default, indicating that mutations in this stacking group should be allowed to stack without restriction. If the policy is set to "f", the first mutation of stacking group at a given site is retained; further mutations of this stacking group at the same site are discarded with no effect. This can be useful for modeling one-way changes; once a gene is disabled by a premature stop codon, for example, you might wish to assume, for simplicity, that further mutations cannot alter that fact. If the policy is set to "l", the last mutation of this stacking group at a given site is retained; earlier mutation of this stacking group at the same site are discarded. This can be useful for modeling an "infinitealleles" scenario in which every new mutation at a site generates a completely new allele, rather than retaining the previous mutations at the site. The mutation stacking policy applies only within the given mutation type's stacking group; mutations of different stacking groups are always allowed to stack in SLiM. The policy applies to all mutations added to the model after the policy is set, whether those mutations are introduced by calls such as addMutation(), addNewMutation(), or addNewDrawnMutation(), or are added by SLiM's own mutation-generation machinery. However, no attempt is made to enforce the policy for mutations already existing at the time the policy is set; typically, therefore, the policy is set in an initialize() callback so that it applies

throughout the simulation. The policy is also not enforced upon the mutations loaded from a file with readFromPopulationFile(); such mutations were governed by whatever stacking policy was in effect when the population file was generated. In nucleotide-based models, the stacking policy for nucleotide-based mutation types is always "l", and cannot be changed. This ensures that new nucleotide mutations always replace the previous nucleotide at a site, and that more than one nucleotide mutation is never present at the same position in a single genome.

nucleotideBased A property of type logical. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** If the mutation type was created with initializeMutationType(), it is not nucleotide-based, and this property is F. If it was created with initializeMutationTypeNuc(), it is nucleotide-based, and this property is T. See those methods for further discussion.

species A property of type Species object. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The species to which the target object belongs.

tag A property of type integer. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** A user-defined integer value. The value of tag is initially undefined, and it is an error to try to read it; if you wish it to have a defined value, you must arrange that yourself by explicitly setting its value prior to using it elsewhere in your code. The value of tag is not used by SLiM; it is free for you to use. See also the getValue() and setValue() methods (provided by the Dictionary class; see the Eidos manual), for another way of attaching state to mutation types.

See Also

Other MutationType: `drawSelectionCoefficient()`, `setDistribution()`

`multiply` *SLiM method multiply*

Description

Documentation for SLiM function `multiply`, which is a method of the SLiM class `SpatialMap`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
multiply(x)
```

Arguments

`x`	An object of type integer or float or SpatialMap object. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 716.

Multiplies the spatial map by x. One possibility is that x is a singleton integer or float value; in this case, each grid value of the target spatial map is multiplied by x. Another possibility is that x is an integer or float vector/matrix/array of the same dimensions as the target spatial map's grid; in this case, each grid value of the target spatial map is multiplied by the corresponding value of x. The third possibility is that x is itself a (singleton) spatial map; in this case, each grid value of the target spatial map is multiplied by the corresponding grid value of x (and thus the two spatial maps must match in their spatiality, their spatial bounds, and their grid dimensions). The target spatial map is returned, to allow easy chaining of operations.

Value

An object of type SpatialMap object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other SpatialMap: `SM`, `add()`, `blend()`, `changeColors()`, `changeValues()`, `divide()`, `exp()`, `gridValues()`, `interpolate()`, `mapColor()`, `mapImage()`, `mapValue()`, `power()`, `range()`, `rescale()`, `sampleImprovedNearbyPoint()`, `sampleNearbyPoint()`, `smooth()`, `subtract()`

`mutation` *SLiM mutation() callback*

Description

This callback specifies that a code block is providing logic to modify mutations. It is called once for each new auto-generated mutation during the tick(s) in which the callback is active. The `mutation()` callback has three possible returns: `T`, `F`, or (beginning in SLiM 3.5) a singleton object of type `Mutation`. see SLiM Manual: page 725

Usage

```
mutation(mut_type_id, subpop_id)
```

Arguments

`mut_type_id`	The id of the mutationType to which this callback should apply. Can be an integer 1, 2, etc., or character ”m1”, ”m2”, etc.
`subpop_id`	The id(s) of the subpopulation(s) to which this callback should apply. Can be an integer 1, 2, etc., or character ”p1”, ”p2”, etc.

Details

Global variables available in reproduction callbacks:

mut The focal mutation that is being modified or reviewed

genome The parental genome that is being copied

element The genomic element that controls the mutation site

originalNuc The nucleotide (0/1/2/3 for A/C/G/T) originally at the mutating position

parent The parent which is generating the offspring genome

subpop The subpopulation in which that individual lives

Value

None

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other callbacks: `early()`, `first()`, `fitnessEffect()`, `fitness()`, `initialize()`, `interaction()`, `late()`, `mateChoice()`, `modifyChild()`, `mutationEffect()`, `recombination()`, `reproduction()`, `slim_callbacks()`, `survival()`

Examples

```
slim_block(recombination(), {
  if (genome1.containsMarkerMutation(m2, 25000) ==
      genome2.containsMarkerMutation(m2, 25000)) {

    return(F)

  }
```

```
  inInv = (breakpoints > 25000) & (breakpoints < 75000)
  if (!any(inInv)) {
    return(F)
  }

  breakpoints = breakpoints[!inInv]
  return(T)

})
```

`mutationCounts` *SLiM method mutationCounts*

Description

Documentation for SLiM function `mutationCounts`, which is a method of the SLiM class `Species`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
mutationCounts(subpops, mutations)
```

Arguments

`subpops`	An object of type null or integer or Subpopulation object. See details for description.
`mutations`	An object of type null or Mutation object. The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 721.

Return an integer vector with the frequency counts of all of the Mutation objects passed in mutations, within the Subpopulation objects in subpops. The subpops argument is required, but you may pass NULL to get population-wide frequency counts. Subpopulations may be supplied either as integer IDs, or as Subpopulation objects. If the optional mutations argument is NULL (the default), frequency counts will be returned for all of the active Mutation objects in the species - the same Mutation objects, and in the same order, as would be returned by the mutations property of sim, in other words. See the -mutationFrequencies() method to obtain float frequencies instead of integer counts. See also the Genome methods mutationCountsInGenomes() and mutationFrequenciesInGenomes().

Value

An object of type integer.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Species: Sp, addSubpopSplit(), addSubpop(), countOfMutationsOfType(), individualsWithPedigreeID killIndividuals(), mutationFrequencies(), mutationsOfType(), outputFixedMutations(), outputFull(), outputMutations(), readFromPopulationFile(), recalculateFitness(), registerFitnessEffectCallback(), registerMateChoiceCallback(), registerModifyChildCallback(), registerMutationCallback(), registerMutationEffectCallback(), registerRecombinationCallback(), registerReproductionCallback(), registerSurvivalCallback(), simulationFinished(), skipTick(), subsetMutations(), treeSeqCoalesced(), treeSeqOutput(), treeSeqRememberIndividuals(), treeSeqSimplify()

---

mutationCountsInGenomes

SLiM method mutationCountsInGenomes

Description

Documentation for SLiM function `mutationCountsInGenomes`, which is a method of the SLiM class `Genome`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
mutationCountsInGenomes(mutations)
```

Arguments

`mutations`	An object of type null or Mutation object. The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 672.

Return an integer vector with the frequency counts of all of the Mutation objects passed in mutations, within the target Genome vector. If the optional mutations argument is NULL (the default), frequency counts will be returned for all of the active Mutation objects in the simulation - the same Mutation objects, and in the same order, as would be returned by the mutations property of sim, in other words. See the +mutationFrequenciesInGenomes() method to obtain float frequencies instead of integer counts. See also the Species methods mutationCounts() and mutationFrequencies(), which may be more efficient for getting counts/frequencies for whole subpopulations or for the whole simulation.

Value

An object of type integer.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Genome: G, addMutations(), addNewDrawnMutation(), addNewMutation(), containsMarkerMutation(), containsMutations(), countOfMutationsOfType(), mutationFrequenciesInGenomes(), mutationsOfType(), nucleotides(), outputMS(), outputVCF(), output(), positionsOfMutationsOfType(), readFromMS(), readFromVCF(), removeMutations(), sumOfMutationsOfType()

---

mutationEffect *SLiM mutationEffect() callback*

Description

A mutationEffect() callback is called by SLiM when it is determining the fitness effect of a mutation carried by an individual. Normally, the fitness effect of a mutation is determined by the selection coefficient s of the mutation and the dominance coefficient h of the mutation (the latter used only if the individual is heterozygous for the mutation). For details on this callback see SLiM Manual: page 712

Usage

```
mutationEffect(mut_type_id, subpop_id)
```

Arguments

`mut_type_id`	The id of the mutationType to which this callback should apply. Can be an integer 1, 2, etc., or character "m1", "m2", etc.
`subpop_id`	The id(s) of the subpopulation(s) to which this callback should apply. Can be an integer 1, 2, etc., or character "p1", "p2", etc.

Details

Global variables available in mutationEffect callbacks:

mut A Mutation object, the mutation whose relative fitness is being evaluated

homozygous A value of T (the mutation is homozygous), F (heterozygous), or NULL (it is paired with a null chromosome, and is thus hemizygous or haploid)

effect The default relative fitness value calculated by SLiM

individual The individual carrying this mutation (an object of class Individual)

subpop The subpopulation in which that individual lives

Value

None

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other callbacks: early(), first(), fitnessEffect(), fitness(), initialize(), interaction(), late(), mateChoice(), modifyChild(), mutation(), recombination(), reproduction(), slim_callbacks(), survival()

Examples

```
slim_block(mutationEffect(), {
  if (homozygous) {
    return(1.0 + mut.selectionCoeff);
  } else {
    return(1.0 + mut.mutationType.dominanceCoeff * mut.selectionCoeff);
  }
})
```

`mutationFrequencies` *SLiM method mutationFrequencies*

Description

Documentation for SLiM function `mutationFrequencies`, which is a method of the SLiM class `Species`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
mutationFrequencies(subpops, mutations)
```

Arguments

`subpops`	An object of type null or integer or Subpopulation object. See details for description.
`mutations`	An object of type null or Mutation object. The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 721.

Return a float vector with the frequencies of all of the Mutation objects passed in mutations, within the Subpopulation objects in subpops. The subpops argument is required, but you may pass NULL to get population-wide frequencies. Subpopulations may be supplied either as integer IDs, or as Subpopulation objects. If the optional mutations argument is NULL (the default), frequencies will be returned for all of the active Mutation objects in the species - the same Mutation objects, and in the same order, as would be returned by the mutations property of sim, in other words. See the -mutationCounts() method to obtain integer counts instead of float frequencies. See also the Genome methods mutationCountsInGenomes() and mutationFrequenciesInGenomes().

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Species: Sp, addSubpopSplit(), addSubpop(), countOfMutationsOfType(), individualsWithPedigreeID
killIndividuals(), mutationCounts(), mutationsOfType(), outputFixedMutations(),
outputFull(), outputMutations(), readFromPopulationFile(), recalculateFitness(),
registerFitnessEffectCallback(), registerMateChoiceCallback(), registerModifyChildCallback(),
registerMutationCallback(), registerMutationEffectCallback(), registerRecombinationCallback(),
registerReproductionCallback(), registerSurvivalCallback(), simulationFinished(),
skipTick(), subsetMutations(), treeSeqCoalesced(), treeSeqOutput(), treeSeqRememberIndividuals(),
treeSeqSimplify()

---

mutationFrequenciesInGenomes

SLiM method mutationFrequenciesInGenomes

Description

Documentation for SLiM function mutationFrequenciesInGenomes, which is a method of the SLiM class Genome. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
mutationFrequenciesInGenomes(mutations)
```

Arguments

mutations	An object of type null or Mutation object. The default value is NULL. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 673.

Return a float vector with the frequencies of all of the Mutation objects passed in mutations, within the target Genome vector. If the optional mutations argument is NULL (the default), frequencies will be returned for all of the active Mutation objects in the simulation - the same Mutation objects, and in the same order, as would be returned by the mutations property of sim, in other words. See the +mutationCountsInGenomes() method to obtain integer counts instead of float frequencies. See also the Species methods mutationCounts() and mutationFrequencies(), which may be more efficient for getting counts/frequencies for whole subpopulations or for the whole simulation.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Genome: G, addMutations(), addNewDrawnMutation(), addNewMutation(), containsMarkerMutation(), containsMutations(), countOfMutationsOfType(), mutationCountsInGenomes(), mutationsOfType(), nucleotides(), outputMS(), outputVCF(), output(), positionsOfMutationsOfType(), readFromMS(), readFromVCF(), removeMutations(), sumOfMutationsOfType()

---

mutationsOfType *SLiM method mutationsOfType*

Description

Documentation for SLiM function `mutationsOfType`, which is a method of the SLiM class `Genome`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Documentation for SLiM function `mutationsOfType`, which is a method of the SLiM class `Species`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
mutationsOfType(mutType)

mutationsOfType(mutType)
```

Arguments

`mutType`	An object of type integer or MutationType object. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 673.

Returns an object vector of all the mutations that are of the type specified by mutType, out of all of the mutations in the genome. If you just need a count of the matching Mutation objects, rather than a vector of the matches, use -countOfMutationsOfType(); if you need just the positions of matching Mutation objects, use -positionsOfMutationsOfType(); and if you are aiming for a sum of the selection coefficients of matching Mutation objects, use -sumOfMutationsOfType(). This method is provided for speed; it is much faster than the corresponding Eidos code.

Documentation for this function can be found in the official SLiM manual: page 721.

Returns an object vector of all the mutations that are of the type specified by mutType, out of all of the mutations that are currently active in the species. If you just need a count of the matching Mutation objects, rather than a vector of the matches, use -countOfMutationsOfType(). This method is often used to look up an introduced mutation at a later point in the simulation, since there is no way to keep persistent references to objects in SLiM. This method is provided for speed; it is much faster than the corresponding Eidos code.

Value

An object of type Mutation object.

An object of type Mutation object.

Copyright

This is documentation for a function in the SLiM software, and has been reproduced from the official manual, which can be found here: http://benhaller.com/slim/SLiM_Manual.pdf. This documentation is Copyright © 2016-2020 Philipp Messer. All rights reserved. More information about SLiM can be found on the official website: https://messerlab.org/slim/

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Genome: G, addMutations(), addNewDrawnMutation(), addNewMutation(), containsMarkerMutation(), containsMutations(), countOfMutationsOfType(), mutationCountsInGenomes(), mutationFrequenciesInGe nucleotides(), outputMS(), outputVCF(), output(), positionsOfMutationsOfType(), readFromMS(), readFromVCF(), removeMutations(), sumOfMutationsOfType()

Other Species: Sp, addSubpopSplit(), addSubpop(), countOfMutationsOfType(), individualsWithPedigreeID
killIndividuals(), mutationCounts(), mutationFrequencies(), outputFixedMutations(),
outputFull(), outputMutations(), readFromPopulationFile(), recalculateFitness(),
registerFitnessEffectCallback(), registerMateChoiceCallback(), registerModifyChildCallback(),
registerMutationCallback(), registerMutationEffectCallback(), registerRecombinationCallback(),
registerReproductionCallback(), registerSurvivalCallback(), simulationFinished(),
skipTick(), subsetMutations(), treeSeqCoalesced(), treeSeqOutput(), treeSeqRememberIndividuals(),
treeSeqSimplify()

`mutationTypesWithIDs` *SLiM method mutationTypesWithIDs*

Description

Documentation for SLiM function `mutationTypesWithIDs`, which is a method of the SLiM class `Community`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
mutationTypesWithIDs(ids)
```

Arguments

`ids`	An object of type integer. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 666.

Find and return the MutationType objects with id values matching the values in ids. If no matching MutationType object can be found with a given id, an error results.

Value

An object of type MutationType object.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Community: Co, createLogFile(), deregisterScriptBlock(), genomicElementTypesWithIDs(), interactionTypesWithIDs(), outputUsage(), registerEarlyEvent(), registerFirstEvent(), registerInteractionCallback(), registerLateEvent(), rescheduleScriptBlock(), scriptBlocksWithIDs(), simulationFinished(), speciesWithIDs(), subpopulationsWithIDs(), usage()

---

nearestInteractingNeighbors

SLiM method nearestInteractingNeighbors

Description

Documentation for SLiM function `nearestInteractingNeighbors`, which is a method of the SLiM class `InteractionType`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
nearestInteractingNeighbors(receiver, count, exerterSubpop, returnDict)
```

Arguments

`receiver`	An object of type Individual object. See details for description.
`count`	An object of type integer. Must be of length 1 (a singleton). The default value is `1`. See details for description.
`exerterSubpop`	An object of type null or Subpopulation object. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`returnDict`	An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 695.

Returns an object<Individual> vector containing up to count interacting individuals that are spatially closest to receiver, according to the distance metric of the InteractionType, from among the exerters in exerterSubpop (or, if that is NULL, then from among all individuals in the receiver's subpopulation). More specifically, this method returns only individuals which can exert an interaction upon receiver, which must be singleton in the default mode of operation (but see below). To obtain all of the interacting individuals within the maximum interaction distance of receiver, simply pass a value for count that is greater than or equal to the size of the exerter subpopulation. Note that if fewer than count interacting individuals are within the maximum interaction distance, the vector returned may be shorter than count, or even zero-length; it is important to check for this possibility even when requesting a single neighbor. If only the number of interacting individuals is

needed, use interactingNeighborCount() instead. The evaluate() method must have been previously called for the receiver and exerter subpopulations, and positions saved at evaluation time will be used. If the InteractionType is non-spatial, this method may not be called. Note that this method uses interaction eligibility as a criterion; it will not return neighbors that cannot exert an interaction upon the receiver (due to the configured receiver or exerter constraints). (It will also never return the receiver as a neighbor of itself.) To find all neighbors of a receiver, whether they can interact with it or not, use nearestNeighbors(). Beginning in SLiM 4.1, this method has a vectorized mode of operation in which the receiver parameter may be non-singleton. To switch the method to this mode, pass T for returnDict, rather than the default of F (the operation of which is described above). In this mode, the return value is a Dictionary object instead of a vector of Individual objects. This dictionary uses integer keys that range from 0 to N-1, where N is the number of individuals passed in receiver; these keys thus correspond directly to the indices of the individuals in receiver, and there is one entry in the dictionary for each receiver. The value in the dictionary, for a given integer key, is an object<Individual> vector with the interacting neighbors found for the corresponding receiver, exactly as described above for the non-vectorized case. The results for each receiver can therefore be obtained from the returned dictionary with getValue(), passing the index of the receiver. The speed of this mode of operation will probably be similar to the speed of making N separate non-vectorized calls to nearestInteractingNeighbors(), but may have other advantages. In this mode of operation, all receivers must belong to the same subpopulation.

Value

An object of type .

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other InteractionType: IT, clippedIntegral(), distanceFromPoint(), distance(), drawByStrength(), evaluate(), interactingNeighborCount(), interactionDistance(), localPopulationDensity(), nearestNeighborsOfPoint(), nearestNeighbors(), neighborCountOfPoint(), neighborCount(), setConstraints(), setInteractionFunction(), strength(), testConstraints(), totalOfNeighborStrengths unevaluate()

`nearestNeighbors` *SLiM method nearestNeighbors*

Description

Documentation for SLiM function `nearestNeighbors`, which is a method of the SLiM class `InteractionType`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
nearestNeighbors(receiver, count, exerterSubpop, returnDict)
```

Arguments

`receiver`	An object of type Individual object. See details for description.
`count`	An object of type integer. Must be of length 1 (a singleton). The default value is `1`. See details for description.
`exerterSubpop`	An object of type null or Subpopulation object. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`returnDict`	An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 696.

Returns an object<Individual> vector containing up to count individuals that are spatially closest to receiver, according to the distance metric of the InteractionType, from among the exerters in exerterSubpop (or, if that is NULL, then from among all individuals in the receiver's subpopulation). In the default mode of operation, receiver must be singleton (but see below). To obtain all of the individuals within the maximum interaction distance of receiver, simply pass a value for count that is greater than or equal to the size of individual's subpopulation. Note that if fewer than count individuals are within the maximum interaction distance, the vector returned may be shorter than count, or even zero-length; it is important to check for this possibility even when requesting a single neighbor. The evaluate() method must have been previously called for the receiver and exerter subpopulations, and positions saved at evaluation time will be used. If the InteractionType is nonspatial, this method may not be called. Note that this method does not use interaction eligibility as a criterion; it will return neighbors that could not interact with the receiver due to the configured receiver or exerter constraints. (It will never return the receiver as a neighbor of itself, however.) To find only neighbors that are eligible to exert an interaction upon the receiver, use nearestInteractingNeighbors(). Beginning in SLiM 4.1, this method has a vectorized mode of operation in which the receiver parameter may be non-singleton. To switch the method to this mode, pass T for returnDict, rather than the default of F

(the operation of which is described above). In this mode, the return value is a Dictionary object instead of a vector of Individual objects. This dictionary uses integer keys that range from 0 to N-1, where N is the number of individuals passed in receiver; these keys thus correspond directly to the indices of the individuals in receiver, and there is one entry in the dictionary for each receiver. The value in the dictionary, for a given integer key, is an object<Individual> vector with the neighbors found for the corresponding receiver, exactly as described above for the non-vectorized case. The results for each receiver can therefore be obtained from the returned dictionary with getValue(), passing the index of the receiver. The speed of this mode of operation will probably be similar to the speed of making N separate non-vectorized calls to nearestNeighbors(), but may have other advantages. In this mode of operation, all receivers must belong to the same subpopulation.

Value

An object of type .

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other InteractionType: IT, clippedIntegral(), distanceFromPoint(), distance(), drawByStrength(), evaluate(), interactingNeighborCount(), interactionDistance(), localPopulationDensity(), nearestInteractingNeighbors(), nearestNeighborsOfPoint(), neighborCountOfPoint(), neighborCount(), setConstraints(), setInteractionFunction(), strength(), testConstraints(), totalOfNeighborStrengths(), unevaluate()

---

nearestNeighborsOfPoint

SLiM method nearestNeighborsOfPoint

Description

Documentation for SLiM function nearestNeighborsOfPoint, which is a method of the SLiM class InteractionType. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
nearestNeighborsOfPoint(point, exerterSubpop, count)
```

Arguments

`point`	An object of type float. See details for description.
`exerterSubpop`	An object of type integer or Subpopulation object. Must be of length 1 (a singleton). See details for description.
`count`	An object of type integer. Must be of length 1 (a singleton). The default value is `1`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 696.

Returns up to count individuals in exerterSubpop that are spatially closest to point, according to the distance metric of the InteractionType. The subpopulation may be supplied either as an integer ID, or as a Subpopulation object. To obtain all of the individuals within the maximum interaction distance of point, simply pass a value for count that is greater than or equal to the size of exerterSubpop. Note that if fewer than count individuals are within the maximum interaction distance, the vector returned may be shorter than count, or even zero-length; it is important to check for this possibility even when requesting a single neighbor. The evaluate() method must have been previously called for exerterSubpop, and positions saved at evaluation time will be used. If the InteractionType is non-spatial, this method may not be called.

Value

An object of type Individual object.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other InteractionType: IT, clippedIntegral(), distanceFromPoint(), distance(), drawByStrength(), evaluate(), interactingNeighborCount(), interactionDistance(), localPopulationDensity(), nearestInteractingNeighbors(), nearestNeighbors(), neighborCountOfPoint(), neighborCount(), setConstraints(), setInteractionFunction(), strength(), testConstraints(), totalOfNeighborStrengths(), unevaluate()

`neighborCount` *SLiM method neighborCount*

Description

Documentation for SLiM function `neighborCount`, which is a method of the SLiM class `InteractionType`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
neighborCount(receivers, exerterSubpop)
```

Arguments

`receivers`	An object of type Individual object. See details for description.
`exerterSubpop`	An object of type null or Subpopulation object. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 696.

Returns the number of neighbors for each individual in receivers, within the maximum interaction distance according to the distance metric of the InteractionType, from among the individuals in exerterSubpop (or, if that is NULL, then from among all individuals in the receiver's subpopulation). All of the receivers must belong to a single subpopulation, and all of the exerters must belong to a single subpopulation, but those two subpopulations do not need to be the same. The evaluate() method must have been previously called for the receiver and exerter subpopulations, and positions saved at evaluation time will be used. This method is similar to nearestNeighbors() (when passed a large count so as to guarantee that all neighbors are returned), but this method returns only a count of the individuals, not a vector containing the individuals. Note that this method does not use interaction eligibility as a criterion; it will count neighbors that cannot exert an interaction upon a receiver (due to the configured receiver or exerter constraints). (It still does not count a receiver as a neighbor of itself, however.) If a count of only interacting neighbors is desired, use interactingNeighborCount(). If the InteractionType is non-spatial, this method may not be called.

Value

An object of type integer.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other InteractionType: IT, clippedIntegral(), distanceFromPoint(), distance(), drawByStrength(), evaluate(), interactingNeighborCount(), interactionDistance(), localPopulationDensity(), nearestInteractingNeighbors(), nearestNeighborsOfPoint(), nearestNeighbors(), neighborCountOfPoint(), setConstraints(), setInteractionFunction(), strength(), testConstraints(), totalOfNeighborStrengths(), unevaluate()

---

neighborCountOfPoint *SLiM method neighborCountOfPoint*

Description

Documentation for SLiM function `neighborCountOfPoint`, which is a method of the SLiM class `InteractionType`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
neighborCountOfPoint(point, exerterSubpop)
```

Arguments

`point`	An object of type float. See details for description.
`exerterSubpop`	An object of type integer or Subpopulation object. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 697.

Returns the number of individuals in exerterSubpop that are within the maximum interaction distance according to the distance metric of the InteractionType. The subpopulation may be supplied either as an integer ID, or as a Subpopulation object. The evaluate()

method must have been previously called for exerterSubpop, and positions saved at evaluation time will be used. If the InteractionType is non-spatial, this method may not be called. This method is similar to nearestNeighborsOfPoint() (when passed a large count so as to guarantee that all neighbors are returned), but this method returns only a count of the individuals, not a vector containing the individuals.

Value

An object of type integer. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other InteractionType: IT, clippedIntegral(), distanceFromPoint(), distance(), drawByStrength(), evaluate(), interactingNeighborCount(), interactionDistance(), localPopulationDensity(), nearestInteractingNeighbors(), nearestNeighborsOfPoint(), nearestNeighbors(), neighborCount(), setConstraints(), setInteractionFunction(), strength(), testConstraints(), totalOfNeighborStrengths(), unevaluate()

---

new_slimr_script_coll

Make a new `slimr_script_coll` *object*

Description

Make a new `slimr_script_coll` object

Usage

```
new_slimr_script_coll(x = list())
```

Arguments

`x`	A list of `slir_script` objects

Value

A `slimr_script_coll` object

`nucleotideCounts` *SLiM method nucleotideCounts*

Description

Documentation for SLiM function `nucleotideCounts`, which is a method of the SLiM class `SLiMBuiltin`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
nucleotideCounts(sequence)
```

Arguments

`sequence`	An object of type integer or string. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 749.

A convenience function that returns an integer vector of length four, providing the number of occurrences of A / C / G / T nucleotides, respectively, in the supplied nucleotide sequence. The parameter sequence may be a singleton string (e.g., "TATA"), a string vector of single characters (e.g., "T", "A", "T", "A"), or an integer vector (e.g., 3, 0, 3, 0), using SLiM's standard code of A=0, C=1, G=2, T=3.

Value

An object of type integer.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other SLiMBuiltin: SB, calcFST(), calcHeterozygosity(), calcInbreedingLoad(), calcPairHeterozygosity(
calcVA(), calcWattersonsTheta(), codonsToAminoAcids(), mm16To256(), mmJukesCantor(),
mmKimura(), nucleotideFrequencies(), nucleotidesToCodons(), summarizeIndividuals(),
treeSeqMetadata()

Examples

```
## This just brings up the documentation:
nucleotideCounts()
```

---

nucleotideFrequencies

SLiM method nucleotideFrequencies

Description

Documentation for SLiM function `nucleotideFrequencies`, which is a method of the SLiM class SLiMBuiltin. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
nucleotideFrequencies(sequence)
```

Arguments

`sequence`	An object of type integer or string. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 749.

A convenience function that returns a float vector of length four, providing the frequencies of occurrences of A / C / G / T nucleotides, respectively, in the supplied nucleotide sequence. The parameter sequence may be a singleton string (e.g., "TATA"), a string vector of single characters 0 0 0 0 0 0 0 0 (e.g., "T", "A", "T", "A"), or an integer vector (e.g., 3, 0, 3, 0), using SLiM's standard code of A=0, C=1, G=2, T=3.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other SLiMBuiltin: SB, calcFST(), calcHeterozygosity(), calcInbreedingLoad(), calcPairHeterozygosity(
calcVA(), calcWattersonsTheta(), codonsToAminoAcids(), mm16To256(), mmJukesCantor(),
mmKimura(), nucleotideCounts(), nucleotidesToCodons(), summarizeIndividuals(),
treeSeqMetadata()

Examples

```
## This just brings up the documentation:
nucleotideFrequencies()
```

---

nucleotides *SLiM method nucleotides*

Description

Documentation for SLiM function `nucleotides`, which is a method of the SLiM class Genome. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
nucleotides(start, end, format)
```

Arguments

start	An object of type null or integer. Must be of length 1 (a singleton). The default value is NULL. See details for description.
end	An object of type null or integer. Must be of length 1 (a singleton). The default value is NULL. See details for description.
format	An object of type string. Must be of length 1 (a singleton). The default value is "string". See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 673.

Returns the nucleotide sequence for the genome. This is the current ancestral sequence, as would be returned by the Chromosome method ancestralNucleotides(), with the nucleotides for any nucleotide-based mutations in the genome overlaid. The range of the returned sequence may be constrained by a start position given in start and/or an end position given in end; nucleotides will be returned from start to end, inclusive. The default value of NULL for start and end represent the first and last base positions of the chromosome, respectively. The format of the returned sequence is controlled by the format parameter. A format of "string" will return the sequence as a singleton string (e.g., "TATA"). A format of "char" will return a string vector with one element per nucleotide (e.g., "T", "A", "T", "A"). A format of "integer" will return an integer vector with values A=0, C=1, G=2, T=3 (e.g., 3, 0, 3, 0). A format of "codon" will return an integer vector with values from 0 to 63, based upon successive nucleotide triplets in the sequence (which, for this format, must have a length that is a multiple of three); see the ancestralNucleotides() documentation for details. If the sequence returned is likely to be long, the "string" format will be the most memory-efficient, and may also be the fastest (but may be harder to work with). Several helper functions are provided for working with sequences, such as nucleotideCounts() to get the counts of A/C/G/T nucleotides in a sequence, nucleotideFrequencies() to get the same information as frequencies, and codonsToAminoAcids() to convert a codon sequence (such as provided by the codon format described above) to an amino acid sequence; see section 25.18.1.

Value

An object of type integer or string.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Genome: G, addMutations(), addNewDrawnMutation(), addNewMutation(), containsMarkerMutation(), containsMutations(), countOfMutationsOfType(), mutationCountsInGenomes(), mutationFrequenciesInGe mutationsOfType(), outputMS(), outputVCF(), output(), positionsOfMutationsOfType(), readFromMS(), readFromVCF(), removeMutations(), sumOfMutationsOfType()

`nucleotidesToCodons` *SLiM method nucleotidesToCodons*

Description

Documentation for SLiM function `nucleotidesToCodons`, which is a method of the SLiM class `SLiMBuiltin`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
nucleotidesToCodons(sequence)
```

Arguments

`sequence`	An object of type integer or string. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 750.

Returns the codon sequence corresponding to the nucleotide sequence in sequence. The codon sequence is an integer vector with values from 0 to 63, based upon successive nucleotide triplets in the nucleotide sequence. The codon value for a given nucleotide triplet XYZ is 16X + 4Y + Z, where X, Y, and Z have the usual values A=0, C=1, G=2, T=3. For example, the triplet AAA has a codon value of 0, AAC is 1, AAG is 2, AAT is 3, ACA is 4, and on upward to TTT which is 63. If the nucleotide sequence AACACATTT is passed in, the codon vector 1 4 63 will therefore be returned. These codon values can be useful in themselves; they can also be passed to codonsToAminoAcids() to translate them into the corresponding amino acid sequence if desired. The nucleotide sequence in sequence may be supplied in any of three formats: a string vector with single-letter nucleotides (e.g., "T", "A", "T", "A"), a singleton string of nucleotide letters (e.g., "TATA"), or an integer vector of nucleotide values (e.g., 3, 0, 3, 0) using SLiM's standard code of A=0, C=1, G=2, T=3. If the choice of format is not driven by other considerations, such as ease of manipulation, then the singleton string format will certainly be the most memory-efficient for long sequences, and will probably also be the fastest. The nucleotide sequence provided must be a multiple of three in length, so that it translates to an integral number of codons. (is)randomNucleotides(integer$ length, [Nif basis = NULL], [string$ format = "string"]) Generates a new random nucleotide sequence with length bases. The four nucleotides ACGT are equally probable if basis is NULL (the default); otherwise, basis may be a 4-element integer or float vector providing relative fractions for A, C, G, and T respectively (these need not sum to 1.0, as they will be normalized). More complex generative models such as Markov processes are not supported intrinsically in SLiM at this time, but arbitrary generated sequences may always be loaded from files on disk. The format parameter controls the format of the returned sequence. It may be "string" to obtain the generated sequence as a singleton string (e.g., "TATA"), "char" to obtain it as a string

vector of single characters (e.g., "T", "A", "T", "A"), or "integer" to obtain it as an integer vector (e.g., 3, 0, 3, 0), using SLiM's standard code of A=0, C=1, G=2, T=3. For passing directly to initializeAncestralNucleotides(), format "string" (a singleton string) will certainly be the most memory-efficient, and probably also the fastest. Memory efficiency can be a significant consideration; the nucleotide sequence for a chromosome of length 109 will occupy approximately 1 GB of memory when stored as a singleton string (with one byte per nucleotide), and much more if stored in the other formats. However, the other formats can be easier to work with in Eidos, and so may be preferable for relatively short chromosomes if you are manipulating the generated sequence.

Value

An object of type integer.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other SLiMBuiltin: SB, calcFST(), calcHeterozygosity(), calcInbreedingLoad(), calcPairHeterozygosity(
calcVA(), calcWattersonsTheta(), codonsToAminoAcids(), mm16To256(), mmJukesCantor(),
mmKimura(), nucleotideCounts(), nucleotideFrequencies(), summarizeIndividuals(),
treeSeqMetadata()

Examples

```
## This just brings up the documentation:
nucleotidesToCodons()
```

---

openDocument *SLiM method openDocument*

Description

Documentation for SLiM function openDocument, which is a method of the SLiM class SLiMgui. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
openDocument(filePath)
```

Arguments

`filePath`	An object of type string. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 711.

Open the document at filePath in SLiMgui, if possible. Supported document types include SLiM model files (typically with a .slim path extension), text files (typically with a .txt path extension, and opened as untitled model files), and PNG, JPG/JPEG, BMP, and GIF image file formats (typically .png / .jpg / .jpeg / .bmp / .gif, respectively). (Note that in SLiMguiLegacy, PDF files (.pdf) are supported but these other image file formats are not.) This method can be particularly useful for opening images created by the simulation itself, often by sublaunching a plotting process in R or another environment; see section 14.8 for an example.

Value

An object of type void or void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other SLiMgui: SG, pauseExecution()

`output` *SLiM method output*

Description

Documentation for SLiM function `output`, which is a method of the SLiM class `Genome`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
output(filePath, append)
```

Arguments

`filePath`	An object of type null or string. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`append`	An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 673.

Output the target genomes in SLiM's native format (see section 27.3.1 for output format details). This low-level output method may be used to output any sample of Genome objects (the Eidos function sample() may be useful for constructing custom samples, as may the SLiM class Individual). For output of a sample from a single Subpopulation, the outputSample() of Subpopulation may be more straightforward to use. If the optional parameter filePath is NULL (the default), output is directed to SLiM's standard output. Otherwise, the output is sent to the file specified by filePath, overwriting that file if append if F, or appending to the end of it if append is T. See outputMS() and outputVCF() for other output formats. Output is generally done in a late() event, so that the output reflects the state of the simulation at the end of a tick.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Genome: G, addMutations(), addNewDrawnMutation(), addNewMutation(), containsMarkerMutation(), containsMutations(), countOfMutationsOfType(), mutationCountsInGenomes(), mutationFrequenciesInGe mutationsOfType(), nucleotides(), outputMS(), outputVCF(), positionsOfMutationsOfType(), readFromMS(), readFromVCF(), removeMutations(), sumOfMutationsOfType()

---

outputFixedMutations *SLiM method outputFixedMutations*

Description

Documentation for SLiM function `outputFixedMutations`, which is a method of the SLiM class Species. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
outputFixedMutations(filePath, append)
```

Arguments

filePath	An object of type null or string. Must be of length 1 (a singleton). The default value is NULL. See details for description.
append	An object of type logical. Must be of length 1 (a singleton). The default value is F. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 721.

Output all fixed mutations - all Substitution objects, in other words (see section 1.5.2) - in a SLiM native format (see section 27.1.2 for output format details). If the optional parameter filePath is NULL (the default), output will be sent to Eidos's output stream (see section 4.2.1). Otherwise, output will be sent to the filesystem path specified by filePath, overwriting that file if append if F, or appending to the end of it if append is T. Mutations which have fixed but have not been turned into Substitution objects - typically because convertToSubstitution has been set to F for their mutation type (see section 25.11.1) - are not output; they are still considered to be segregating mutations by SLiM. In SLiM 3.3 and later, the output format includes the nucleotides associated with any nucleotide-based mutations; see section 27.1.2. Output is generally done in a late() event, so that the output reflects the state of the simulation at the end of a tick.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Species: Sp, addSubpopSplit(), addSubpop(), countOfMutationsOfType(), individualsWithPedigreeID killIndividuals(), mutationCounts(), mutationFrequencies(), mutationsOfType(), outputFull(), outputMutations(), readFromPopulationFile(), recalculateFitness(), registerFitnessEffectCallback(), registerMateChoiceCallback(), registerModifyChildCallback(), registerMutationCallback(), registerMutationEffectCallback(), registerRecombinationCallback(), registerReproductionCallback(), registerSurvivalCallback(), simulationFinished(), skipTick(), subsetMutations(), treeSeqCoalesced(), treeSeqOutput(), treeSeqRememberIndividuals(), treeSeqSimplify()

---

outputFull *SLiM method outputFull*

Description

Documentation for SLiM function `outputFull`, which is a method of the SLiM class Species. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
outputFull(
  filePath,
  binary,
  append,
  spatialPositions,
  ages,
  ancestralNucleotides,
  pedigreeIDs
)
```

Arguments

`filePath`	An object of type null or string. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`binary`	An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.
`append`	An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.
`spatialPositions`	An object of type logical. Must be of length 1 (a singleton). The default value is `T`. See details for description.
`ages`	An object of type logical. Must be of length 1 (a singleton). The default value is `T`. See details for description.
`ancestralNucleotides`	An object of type logical. Must be of length 1 (a singleton). The default value is `T`. See details for description.
`pedigreeIDs`	An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 722.

Output the state of the entire population (see section 27.1.1 for output format details). If the optional parameter filePath is NULL (the default), output will be sent to Eidos's output stream (see section 4.2.1). Otherwise, output will be sent to the filesystem path specified by filePath, overwriting that file if append if F, or appending to the end of it if append is T. When writing to a file, a logical flag, binary, may be supplied as well. If binary is T, the population state will be written as a binary file instead of a text file (binary data cannot be written to the standard output stream). The binary file is usually smaller, and in any case will be read much faster than the corresponding text file would be read. Binary files are not guaranteed to be portable between platforms; in other words, a binary file written on one machine may not be readable on a different machine (but in practice it usually will be, unless the platforms being used are fairly unusual). If binary is F (the default), a text file will be written. Beginning with SLiM 2.3, the spatialPositions parameter may be used to control the output of the spatial positions of individuals in species for which continuous space has been enabled using the dimensionality option of initializeSLiMOptions(). If spatialPositions is F, the output will not contain spatial positions, and will be identical to the output generated by SLiM 2.1 and later. If spatialPositions is T, spatial position information will be output if it is available (see section 27.1.1 for format details). If the species does not have continuous space enabled, the spatialPositions parameter will be ignored. Positional information may be output for all output destinations - the Eidos output stream, a text file, or a binary file. Beginning with SLiM 3.0, the ages parameter may be used to control the output of the ages of individuals in nonWF simulations. If ages is F, the output will not contain ages, preserving backward compatibility with the output format of SLiM 2.1 and later. If ages is T, ages will be output for nonWF models (see section 27.1.1 for format details). In WF simulations, the ages parameter will be ignored. Beginning with SLiM 3.3, the ancestralNucleotides parameter may be used to control the output of

the ancestral nucleotide sequence in nucleotide-based models (see section 27.1.1 for format details). If ancestralNucleotides is F, the output will not contain ancestral nucleotide information, and so the ancestral sequence will not be restored correctly if the saved file is loaded with readPopulationFile(). This option is provided because the ancestral sequence may be quite large, for models with a long chromosome (e.g., 1 GB if the chromosome is 109 bases long, when saved in text format, or 0.25 GB when saved in binary format). If the model is not nucleotide-based (as enabled with the nucleotideBased parameter to initializeSLiMOptions()), the ancestralNucleotides parameter will be ignored. Note that in nucleotide-based models the output format will always include the nucleotides associated with any nucleotide-based mutations; the ancestralNucleotides flag governs only the ancestral sequence. Beginning with SLiM 3.5, the pedigreeIDs parameter may be used to request that pedigree IDs be written out (and read in by readFromPopulationFile(), subsequently). This option is turned off (F) by default, to preserve backward compatibility; if it is turned on (T), different file version values will be used, and backward compatibility with previous versions of SLiM will be lost (see section 27.1.1). This option may only be used if SLiM's optional pedigree tracking has been enabled with initializeSLiMOptions(keepPedigrees=T). Output is generally done in a late() event, so that the output reflects the state of the simulation at the end of a tick.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Species: Sp, addSubpopSplit(), addSubpop(), countOfMutationsOfType(), individualsWithPedigreeID
killIndividuals(), mutationCounts(), mutationFrequencies(), mutationsOfType(),
outputFixedMutations(), outputMutations(), readFromPopulationFile(), recalculateFitness(),
registerFitnessEffectCallback(), registerMateChoiceCallback(), registerModifyChildCallback(),
registerMutationCallback(), registerMutationEffectCallback(), registerRecombinationCallback(),
registerReproductionCallback(), registerSurvivalCallback(), simulationFinished(),
skipTick(), subsetMutations(), treeSeqCoalesced(), treeSeqOutput(), treeSeqRememberIndividuals(),
treeSeqSimplify()

`outputMS` *SLiM method outputMS*

Description

Documentation for SLiM function `outputMS`, which is a method of the SLiM class `Genome`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
outputMS(filePath, append, filterMonomorphic)
```

Arguments

`filePath`	An object of type null or string. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`append`	An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.
`filterMonomorphic`	An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 674.

Output the target genomes in MS format (see section 27.3.2 for output format details). This low-level output method may be used to output any sample of Genome objects (the Eidos function sample() may be useful for constructing custom samples, as may the SLiM class Individual). For output of a sample from a single Subpopulation, the outputMSSample() of Subpopulation may be more straightforward to use. If the optional parameter filePath is NULL (the default), output is directed to SLiM's standard output. Otherwise, the output is sent to the file specified by filePath, overwriting that file if append if F, or appending to the end of it if append is T. Positions in the output will span the interval [0,1]. If filterMonomorphic is F (the default), all mutations that are present in the sample will be included in the output. This means that some mutations may be included that are actually monomorphic within the sample (i.e., that exist in every sampled genome, and are thus apparently fixed). These may be filtered out with filterMonomorphic = T if desired; note that this option means that some mutations that do exist in the sampled genomes might not be included in the output, simply because they exist in every sampled genome. See output() and outputVCF() for other output formats. Output is generally done in a late() event, so that the output reflects the state of the simulation at the end of a tick.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Genome: G, addMutations(), addNewDrawnMutation(), addNewMutation(), containsMarkerMutation(), containsMutations(), countOfMutationsOfType(), mutationCountsInGenomes(), mutationFrequenciesInGe mutationsOfType(), nucleotides(), outputVCF(), output(), positionsOfMutationsOfType(), readFromMS(), readFromVCF(), removeMutations(), sumOfMutationsOfType()

---

outputMSSample *SLiM method outputMSSample*

Description

Documentation for SLiM function outputMSSample, which is a method of the SLiM class Subpopulation. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
outputMSSample(
  sampleSize,
  replace,
  requestedSex,
  filePath,
  append,
  filterMonomorphic
)
```

Arguments

sampleSize	An object of type integer. Must be of length 1 (a singleton). See details for description.
replace	An object of type logical. Must be of length 1 (a singleton). The default value is T. See details for description.

`requestedSex` An object of type string. Must be of length 1 (a singleton). The default value is `"*"`. See details for description.

`filePath` An object of type null or string. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

`append` An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

`filterMonomorphic` An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 740.

Output a random sample from the subpopulation in MS format (see section 27.2.2 for output format details). Positions in the output will span the interval [0,1]. A sample of genomes (not entire individuals, note) of size sampleSize from the subpopulation will be output. The sample may be done either with or without replacement, as specified by replace; the default is to sample with replacement. A particular sex of individuals may be requested for the sample, for simulations in which sex is enabled, by passing "M" or "F" for requestedSex; passing "*", the default, indicates that genomes from individuals should be selected randomly, without respect to sex. If the sampling options provided by this method are not adequate, see the outputMS() method of Genome for a more flexible low-level option. If the optional parameter filePath is NULL (the default), output will be sent to Eidos's output stream (see section 4.2.1). Otherwise, output will be sent to the filesystem path specified by filePath, overwriting that file if append if F, or appending to the end of it if append is T. If filterMonomorphic is F (the default), all mutations that are present in the sample will be included in the output. This means that some mutations may be included that are actually monomorphic within the sample (i.e., that exist in every sampled genome, and are thus apparently fixed). These may be filtered out with filterMonomorphic = T if desired; note that this option means that some mutations that do exist in the sampled genomes might not be included in the output, simply because they exist in every sampled genome. See outputSample() and outputVCFSample() for other output formats. Output is generally done in a late() event, so that the output reflects the state of the simulation at the end of a tick.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Subpopulation: P, addCloned(), addCrossed(), addEmpty(), addRecombinant(), addSelfed(), addSpatialMap(), cachedFitness(), configureDisplay(), defineSpatialMap(), outputSample(), outputVCFSample(), pointDeviated(), pointInBounds(), pointPeriodic(), pointReflected(), pointStopped(), pointUniform(), removeSpatialMap(), removeSubpopulation(), sampleIndividuals(), setCloningRate(), setMigrationRates(), setSelfingRate(), setSexRatio(), setSpatialBounds(), setSubpopulationSize(), spatialMapColor(), spatialMapImage(), spatialMapValue(), subsetIndividuals(), takeMigrants()

---

outputMutations *SLiM method outputMutations*

Description

Documentation for SLiM function `outputMutations`, which is a method of the SLiM class Species. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
outputMutations(mutations, filePath, append)
```

Arguments

`mutations`	An object of type Mutation object. See details for description.
`filePath`	An object of type null or string. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`append`	An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 723.

Output all of the given mutations (see section 27.1.3 for output format details). This can be used to output all mutations of a given mutation type, for example. If the optional parameter filePath is NULL (the default), output will be sent to Eidos's output stream (see section 4.2.1). Otherwise, output will be sent to the filesystem path specified by filePath, overwriting that file if append if F, or appending to the end of it if append is T. In SLiM 3.3 and later, the output format includes the nucleotides associated with any nucleotide-based mutations; see section 27.1.3. Output is generally done in a late() event, so that the output reflects the state of the simulation at the end of a tick.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Species: Sp, addSubpopSplit(), addSubpop(), countOfMutationsOfType(), individualsWithPedigreeID killIndividuals(), mutationCounts(), mutationFrequencies(), mutationsOfType(), outputFixedMutations(), outputFull(), readFromPopulationFile(), recalculateFitness(), registerFitnessEffectCallback(), registerMateChoiceCallback(), registerModifyChildCallback(), registerMutationCallback(), registerMutationEffectCallback(), registerRecombinationCallback(), registerReproductionCallback(), registerSurvivalCallback(), simulationFinished(), skipTick(), subsetMutations(), treeSeqCoalesced(), treeSeqOutput(), treeSeqRememberIndividuals(), treeSeqSimplify()

---

outputSample *SLiM method outputSample*

Description

Documentation for SLiM function `outputSample`, which is a method of the SLiM class Subpopulation. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
outputSample(sampleSize, replace, requestedSex, filePath, append)
```

Arguments

`sampleSize`	An object of type integer. Must be of length 1 (a singleton). See details for description.
`replace`	An object of type logical. Must be of length 1 (a singleton). The default value is `T`. See details for description.

`requestedSex`	An object of type string. Must be of length 1 (a singleton). The default value is `"*"`. See details for description.
`filePath`	An object of type null or string. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`append`	An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 740.

Output a random sample from the subpopulation in SLiM's native format (see section 27.2.1 for output format details). A sample of genomes (not entire individuals, note) of size sampleSize from the subpopulation will be output. The sample may be done either with or without replacement, as specified by replace; the default is to sample with replacement. A particular sex of individuals may be requested for the sample, for simulations in which sex is enabled, by passing "M" or "F" for requestedSex; passing "*", the default, indicates that genomes from individuals should be selected randomly, without respect to sex. If the sampling options provided by this method are not adequate, see the output() method of Genome for a more flexible low-level option. If the optional parameter filePath is NULL (the default), output will be sent to Eidos's output stream (see section 4.2.1). Otherwise, output will be sent to the filesystem path specified by filePath, overwriting that file if append if F, or appending to the end of it if append is T. See outputMSSample() and outputVCFSample() for other output formats. Output is generally done in a late() event, so that the output reflects the state of the simulation at the end of a tick.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Subpopulation: `P`, `addCloned()`, `addCrossed()`, `addEmpty()`, `addRecombinant()`, `addSelfed()`, `addSpatialMap()`, `cachedFitness()`, `configureDisplay()`, `defineSpatialMap()`, `outputMSSample()`, `outputVCFSample()`, `pointDeviated()`, `pointInBounds()`, `pointPeriodic()`, `pointReflected()`, `pointStopped()`, `pointUniform()`, `removeSpatialMap()`, `removeSubpopulation()`, `sampleIndividuals()`, `setCloningRate()`, `setMigrationRates()`, `setSelfingRate()`,

setSexRatio(), setSpatialBounds(), setSubpopulationSize(), spatialMapColor(), spatialMapImage(), spatialMapValue(), subsetIndividuals(), takeMigrants()

---

outputUsage *SLiM method outputUsage*

Description

Documentation for SLiM function outputUsage, which is a method of the SLiM class Community. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
outputUsage(void)
```

Arguments

void	An object of type . See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 666.

Output the current memory usage of the simulation to Eidos's output stream. The specifics of what is printed, and in what format, should not be relied upon as they may change from version to version of SLiM. This method is primarily useful for understanding where the memory usage of a simulation predominantly resides, for debugging or optimization. Note that it does not capture all memory usage by the process; rather, it summarizes the memory usage by SLiM and Eidos in directly allocated objects and buffers. To get the same memory usage reported by outputUsage(), but as a float$ value, use the Community method usage(). To get the total memory usage of the running process (either current or peak), use the Eidos function usage().

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Community: Co, createLogFile(), deregisterScriptBlock(), genomicElementTypesWithIDs(), interactionTypesWithIDs(), mutationTypesWithIDs(), registerEarlyEvent(), registerFirstEvent(), registerInteractionCallback(), registerLateEvent(), rescheduleScriptBlock(), scriptBlocksWithIDs(), simulationFinished(), speciesWithIDs(), subpopulationsWithIDs(), usage()

---

outputVCF *SLiM method outputVCF*

Description

Documentation for SLiM function outputVCF, which is a method of the SLiM class Genome. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
outputVCF(
  filePath,
  outputMultiallelics,
  append,
  simplifyNucleotides,
  outputNonnucleotides
)
```

Arguments

filePath	An object of type null or string. Must be of length 1 (a singleton). The default value is NULL. See details for description.
outputMultiallelics	An object of type logical. Must be of length 1 (a singleton). The default value is T. See details for description.
append	An object of type logical. Must be of length 1 (a singleton). The default value is F. See details for description.
simplifyNucleotides	An object of type logical. Must be of length 1 (a singleton). The default value is F. See details for description.
outputNonnucleotides	An object of type logical. Must be of length 1 (a singleton). The default value is T. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 674.

Output the target genomes in VCF format (see sections 27.2.3, 27.2.4, and 27.3.3 for output format details). The target genomes are treated as pairs comprising individuals for purposes of structuring the VCF output, so an even number of genomes is required. This low-level output method may be used to output any sample of Genome objects (the Eidos function sample() may be useful for constructing custom samples, as may the SLiM class Individual). For output of a sample from a single Subpopulation, the outputVCFSample() of Subpopulation may be more straightforward to use. If the optional parameter filePath is NULL (the default), output is directed to SLiM's standard output. Otherwise, the output is sent to the file specified by filePath, overwriting that file if append if F, or appending to the end of it if append is T. The parameters outputMultiallelics, simplifyNucleotides, and outputNonnucleotides affect the format of the output produced; see sections 27.2.3 and 27.2.4 for further discussion. See outputMS() and output() for other output formats. Output is generally done in a late() event, so that the output reflects the state of the simulation at the end of a tick.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Genome: G, addMutations(), addNewDrawnMutation(), addNewMutation(), containsMarkerMutation(), containsMutations(), countOfMutationsOfType(), mutationCountsInGenomes(), mutationFrequenciesInGe mutationsOfType(), nucleotides(), outputMS(), output(), positionsOfMutationsOfType(), readFromMS(), readFromVCF(), removeMutations(), sumOfMutationsOfType()

`outputVCFSample` *SLiM method outputVCFSample*

Description

Documentation for SLiM function `outputVCFSample`, which is a method of the SLiM class `Subpopulation`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
outputVCFSample(
  sampleSize,
  replace,
  requestedSex,
  outputMultiallelics,
  filePath,
  append,
  simplifyNucleotides,
  outputNonnucleotides
)
```

Arguments

`sampleSize`	An object of type integer. Must be of length 1 (a singleton). See details for description.
`replace`	An object of type logical. Must be of length 1 (a singleton). The default value is `T`. See details for description.
`requestedSex`	An object of type string. Must be of length 1 (a singleton). The default value is `"*"`. See details for description.
`outputMultiallelics`	An object of type logical. Must be of length 1 (a singleton). The default value is `T`. See details for description.
`filePath`	An object of type null or string. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`append`	An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.
`simplifyNucleotides`	An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.
`outputNonnucleotides`	An object of type logical. Must be of length 1 (a singleton). The default value is `T`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 741.

Output a random sample from the subpopulation in VCF format (see sections 27.2.3 and 27.2.4 for output format details). A sample of individuals (not genomes, note - unlike

the outputSample() and outputMSSample() methods) of size sampleSize from the subpopulation will be output. The sample may be done either with or without replacement, as specified by replace; the default is to sample with replacement. A particular sex of individuals may be requested for the sample, for simulations in which sex is enabled, by passing "M" or "F" for requestedSex; passing "*", the default, indicates that genomes from individuals should be selected randomly, without respect to sex. If the sampling options provided by this method are not adequate, see the outputVCF() method of Genome for a more flexible low-level option. If the optional parameter filePath is NULL (the default), output will be sent to Eidos's output stream (see section 4.2.1). Otherwise, output will be sent to the filesystem path specified by filePath, overwriting that file if append if F, or appending to the end of it if append is T. The parameters outputMultiallelics, simplifyNucleotides, and outputNonnucleotides affect the format of the output produced; see sections 27.2.3 and 27.2.4 for further discussion. See outputMSSample() and outputSample() for other output formats. Output is generally done in a late() event, so that the output reflects the state of the simulation at the end of a tick.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Subpopulation: P, addCloned(), addCrossed(), addEmpty(), addRecombinant(), addSelfed(), addSpatialMap(), cachedFitness(), configureDisplay(), defineSpatialMap(), outputMSSample(), outputSample(), pointDeviated(), pointInBounds(), pointPeriodic(), pointReflected(), pointStopped(), pointUniform(), removeSpatialMap(), removeSubpopulation(), sampleIndividuals(), setCloningRate(), setMigrationRates(), setSelfingRate(), setSexRatio(), setSpatialBounds(), setSubpopulationSize(), spatialMapColor(), spatialMapImage(), spatialMapValue(), subsetIndividuals(), takeMigrants()

P *Subpopulation*

Description

Documentation for Subpopulation class from SLiM

Details

This class represents one subpopulation in the simulated population. Section 1.5.5 presents an overview of the conceptual role of this class. The subpopulations currently defined in the simulation are defined as global constants with the same names used in the SLiM input file - p1, p2, and so forth. This class has the following methods (functions):

- `addCloned`
- `addCrossed`
- `addEmpty`
- `addRecombinant`
- `addSelfed`
- `addSpatialMap`
- `cachedFitness`
- `configureDisplay`
- `defineSpatialMap`
- `outputMSSample`
- `outputSample`
- `outputVCFSample`
- `pointDeviated`
- `pointInBounds`
- `pointPeriodic`
- `pointReflected`
- `pointStopped`
- `pointUniform`
- `removeSpatialMap`
- `removeSubpopulation`
- `sampleIndividuals`
- `setCloningRate`
- `setMigrationRates`
- `setSelfingRate`
- `setSexRatio`
- `setSpatialBounds`
- `setSubpopulationSize`
- `spatialMapColor`
- `spatialMapImage`
- `spatialMapValue`
- `subsetIndividuals`
- `takeMigrants`

This class has the following properties:

cloningRate A property of type float. This property is a constant, so it is not modifiable. **Property Description:** The fraction of children in the next generation that will be produced by cloning (as opposed to biparental mating). In non-sexual (i.e. hermaphroditic) simulations, this property is a singleton float representing the overall subpopulation cloning rate. In sexual simulations, this property is a float vector with two values: the cloning rate for females (at index 0) and for males (at index 1).

description A property of type string. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** A human-readable string description for the subpopulation. By default, this is the empty string, ""; however, it may be set to whatever you wish. When tree-sequence recording is enabled, description is persisted in the subpopulation's metadata in tree-sequence output.

firstMaleIndex A property of type integer. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The index of the first male individual in the subpopulation. The genomes vector is sorted into females first and males second; firstMaleIndex gives the position of the boundary between those sections. Note, however, that there are two genomes per diploid individual, and the firstMaleIndex is not premultiplied by 2; you must multiply it by 2 before using it to decide whether a given index into genomes is a genome for a male or a female. The firstMaleIndex property is also the number of females in the subpopulation, given this design. For non-sexual (i.e. hermaphroditic) simulations, this property has an undefined value and should not be used.

fitnessScaling A property of type float. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** A float scaling factor applied to the fitness of all individuals in this subpopulation (i.e., the fitness value computed for each individual will be multiplied by this value). This is primarily of use in nonWF models, where fitness is absolute, rather than in WF models, where fitness is relative (and thus a constant factor multiplied into the fitness of every individual will make no difference); however, it may be used in either type of model. This provides a simple, fast way to modify the fitness of all individuals in a subpopulation; conceptually it is similar to returning the same fitness effect for all individuals in the subpopulation from a fitnessEffect() callback, but without the complexity and performance overhead of implementing such a callback. To scale the fitness of individuals by different (individual-specific) factors, see the fitnessScaling property of Individual. The value of fitnessScaling is reset to 1.0 every tick, so that any scaling factor set lasts for only a single tick. This reset occurs immediately after fitness values are calculated, in both WF and nonWF models.

genomes A property of type Genome object. This property is a constant, so it is not modifiable. **Property Description:** All of the genomes contained by the subpopulation; there are two genomes per diploid individual.

genomesNonNull A property of type Genome object. This property is a constant, so it is not modifiable. **Property Description:** All of the genomes contained by the subpopulation, as with the genomes property, if all of them are not null genomes; any null genomes present are excluded from the returned vector. This is a convenience shorthand, sometimes useful in models that involve null genomes.

id A property of type integer. It is of length one (a singleton). This property is a constant,

so it is not modifiable. **Property Description:** The identifier for this subpopulation; for subpopulation p3, for example, this is 3.

immigrantSubpopFractions A property of type float. This property is a constant, so it is not modifiable. **Property Description:** The expected value of the fraction of children in the next generation that are immigrants arriving from particular subpopulations.

immigrantSubpopIDs A property of type integer. This property is a constant, so it is not modifiable. **Property Description:** The identifiers of the particular subpopulations from which immigrants will arrive in the next generation.

individualCount A property of type integer. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The number of individuals in the subpopulation; one-half of the number of genomes.

individuals A property of type Individual object. This property is a constant, so it is not modifiable. **Property Description:** All of the individuals contained by the subpopulation. Each individual is diploid and thus contains two Genome objects. See the sampleIndividuals() and subsetIndividuals() for fast ways to get a subset of the individuals in a subpopulation.

lifetimeReproductiveOutput A property of type integer. This property is a constant, so it is not modifiable. **Property Description:** If pedigree tracking is turned on with initializeSLiMOptions(keepPedigrees=T), lifetimeReproductiveOutput contains the value of the Individual property reproductiveOutput for all individuals in the subpopulation that died in the last viability/survival tick cycle stage (or, for WF models, immediately after reproduction). This allows access to the lifetime reproductive output of individuals in the subpopulation at the end of their lives. If pedigree tracking is not on, this property is unavailable.

lifetimeReproductiveOutputF A property of type integer. This property is a constant, so it is not modifiable. **Property Description:** If pedigree tracking is turned on with initializeSLiMOptions(keepPedigrees=T), lifetimeReproductiveOutputF contains the value of the Individual property reproductiveOutput for all female individuals in the subpopulation that died in the last viability/ survival tick cycle stage (or, for WF models, immediately after reproduction). This property is undefined if separate sexes have not been enabled, or if pedigree tracking is not on.

lifetimeReproductiveOutputM A property of type integer. This property is a constant, so it is not modifiable. **Property Description:** If pedigree tracking is turned on with initializeSLiMOptions(keepPedigrees=T), lifetimeReproductiveOutputM contains the value of the Individual property reproductiveOutput for all male individuals in the subpopulation that died in the last viability/ survival tick cycle stage (or, for WF models, immediately after reproduction). This property is undefined if separate sexes have not been enabled, or if pedigree tracking is not on.

name A property of type string. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** A human-readable string name for the subpopulation. By default, this is the subpopulation's symbol as a string; for subpopulation p3, for example, name defaults to "p3". However, it may be set to whatever you wish except that subpopulation names must be unique across time (two different subpopulations may not both have the name "foo", even if they never exist at the same time). A subpopulation's name may appear as a label in SLiMgui, and it can be useful in generating output, debugging, and other purposes. When tree-sequence recording is enabled, name is persisted in the subpopulation's metadata in

tree-sequence output, and can then be used in Python to identify the subpopulation; if you plan to take advantage of that feature, name should follow the syntax of Python identifiers: starting with a letter or underscore [a-zA-Z_], followed by letters, digits, or underscores [a-zA-Z0-9_], without spaces, hyphens, or other characters.

selfingRate A property of type float. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The expected value of the fraction of children in the next generation that will be produced by selfing (as opposed to biparental mating). Selfing is only possible in non-sexual (i.e. hermaphroditic) simulations; for sexual simulations this property always has a value of 0.0.

sexRatio A property of type float. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** For sexual simulations, the sex ratio for the subpopulation. This is defined, in SLiM, as the fraction of the subpopulation that is male; in other words, it is actually the M:(M+F) ratio. For non-sexual (i.e. hermaphroditic) simulations, this property has an undefined value and should not be used.

spatialBounds A property of type float. This property is a constant, so it is not modifiable. **Property Description:** The spatial boundaries of the subpopulation. The length of the spatialBounds property depends upon the spatial dimensionality declared with initializeSLiMOptions(). If the spatial dimensionality is zero (as it is by default), the value of this property is float(0) (a zero-length float vector). Otherwise, minimums are supplied for each coordinate used by the dimensionality of the simulation, followed by maximums for each. In other words, if the declared dimensionality is "xy", the spatialBounds property will contain values (x0, y0, x1, y1); bounds for the z coordinate will not be included in that case, since that coordinate is not used in the simulation's dimensionality. This property cannot be set, but the setSpatialBounds() method may be used to achieve the same thing.

spatialMaps A property of type SpatialMap object. This property is a constant, so it is not modifiable. **Property Description:** The spatial maps that are currently added to the subpopulation.

species A property of type Species object. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The species to which the target object belongs.

tag A property of type integer. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** A user-defined integer value. The value of tag is initially undefined, and it is an error to try to read it; if you wish it to have a defined value, you must arrange that yourself by explicitly setting its value prior to using it elsewhere in your code. The value of tag is not used by SLiM; it is free for you to use. See also the getValue() and setValue() methods (provided by the Dictionary class; see the Eidos manual), for another way of attaching state to subpopulations.

See Also

Other Subpopulation: `addCloned()`, `addCrossed()`, `addEmpty()`, `addRecombinant()`, `addSelfed()`, `addSpatialMap()`, `cachedFitness()`, `configureDisplay()`, `defineSpatialMap()`, `outputMSSample()`, `outputSample()`, `outputVCFSample()`, `pointDeviated()`, `pointInBounds()`, `pointPeriodic()`, `pointReflected()`, `pointStopped()`, `pointUniform()`, `removeSpatialMap()`, `removeSubpopulation()`, `sampleIndividuals()`, `setCloningRate()`, `setMigrationRates()`, `setSelfingRate()`,

`setSexRatio()`, `setSpatialBounds()`, `setSubpopulationSize()`, `spatialMapColor()`, `spatialMapImage()`, `spatialMapValue()`, `subsetIndividuals()`, `takeMigrants()`

`pauseExecution` *SLiM method pauseExecution*

Description

Documentation for SLiM function `pauseExecution`, which is a method of the SLiM class `SLiMgui`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
pauseExecution(filePath)
```

Arguments

`filePath`	An object of type string. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 711.

Pauses a model that is playing in SLiMgui. This is essentially equivalent to clicking the "Play" button to stop the execution of the model. Execution can be resumed by the user, by clicking the "Play" button again; unlike calling stop() or simulationFinished(), the simulation is not terminated. This method can be useful for debugging or exploratory purposes, to pause the model at a point of interest. Execution is paused at the end of the currently executing tick, not mid-tick. If the model is being profiled, or is executing forward to a tick number entered in the tick field, pauseExecution() will do nothing; by design, pauseExecution() only pauses execution when SLiMgui is doing a simple "Play" of the model.

Value

An object of type void or void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other SLiMgui: SG, openDocument()

`pointDeviated` *SLiM method pointDeviated*

Description

Documentation for SLiM function `pointDeviated`, which is a method of the SLiM class Subpopulation. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
pointDeviated(n, point, boundary, maxDistance, functionType, ...)
```

Arguments

`n`	An object of type integer. Must be of length 1 (a singleton). See details for description.
`point`	An object of type float. See details for description.
`boundary`	An object of type string. Must be of length 1 (a singleton). See details for description.
`maxDistance`	An object of type numeric. Must be of length 1 (a singleton). See details for description.
`functionType`	An object of type string. Must be of length 1 (a singleton). See details for description.
`...`	An object of type NA. NA See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 741.

Returns a vector containing n points that are derived from point by adding a deviation drawn from a dispersal kernel (specified by maxDistance, functionType, and the ellipsis parameters ..., as detailed below) and then applying a boundary condition specified by boundary. This method therefore performs the steps of a simple dispersal algorithm in a single vectorized call. The parameter point may contain a single point which is deviated and bounded n independent times, or may contain n points each of which is deviated and bounded. In any case, each point in point should match the dimensionality of the model -

one element in a 1D model, two elements in a 2D model, or three elements in a 3D model. This method should not be called in a non-spatial model. The dispersal kernel is specified similarly to other kernel-based methods, such as setInteractionFunction() and smooth(). For pointDeviated(), functionType may be "f" with no ellipsis arguments ... to use a flat kernel out to maxDistance; "l" with no ellipsis arguments for a kernel that decreases linearly from the center to zero at maxDistance; "e", in which case the ellipsis should supply a numeric$ lambda (rate) parameter for a negative exponential function; "n", in which case the ellipsis should supply a numeric$ sigma (standard deviation) parameter for a Gaussian function; or "t", in which case the ellipsis should supply a numeric$ degrees of freedom and a numeric$ scale parameter for a t-distribution function. The Cauchy ("c") kernel is not supported by pointDeviated() since it is not well-behaved for this purpose, and the Student's t ("t") kernel is not allowed in 3D models at present simply because it hasn't been implemented. See the InteractionType class documentation (section 25.8) for more detailed discussion of the available kernel types and their parameters and probability distribution functions. The random points returned from this method are drawn from the probability distribution that is radially symmetric and has density proportional to the kernel - in other words, at distance r the density is proportional to the kernel type referred to by functionType. (Said another way, the shape of the cross-section through the probability density function is given by the kernel.) For instance, the value of the type "e" (exponential) kernel with rate a at r is proportional to exp(−ar), and so in 2D, the probability density that this method with kernel type "e" draws from has density proportional to p(x, y) = exp(−a sqrt(x2 + y2)), since r = sqrt(x2 + y2) is the distance. Note that the distribution of the distance is not given by the kernel except in 1D: in the type "e" example, the distribution of the distance in 1D is exponential, while in 2D it has density proportional to r exp(−ar) (i.e., Gamma with shape parameter 1). For another example, the value of the type "n" (Normal) kernel at r with standard deviation 1 is proportional to exp(−r2 / 2), and so the density is proportional to p(x, y) = exp(− (x2 + y2) / 2). This is the standard bivariate Normal, and equivalent to drawing independent Normals for the x and y directions; however, the Normal is the only distribution for which independent draws along each axis will result in a radially symmetric distribution. The distribution of the distance in 2D with type "n" is proportional to r exp(−r2 / 2), i.e., Rayleigh. The boundary condition must be one of "none", "periodic", "reflecting", "stopping", or "reprising". For "none", no boundary condition is enforced; the deviated points are simply returned as is. For "periodic", "reflecting", and "stopping", the boundary condition is enforced just as it is by the pointPeriodic(), pointReflected(), and pointStopped() methods; see their documentation for further details. For "reprising", if the deviated point is out of bounds a new deviated point will be chosen, based upon the same original point, until a point inside bounds is obtained. Note that absorbing boundaries (for which being out-of-bounds is lethal) would be implemented in script; this method cannot enforce them. In the typical usage case, point comes from the spatialPosition property for a vector of individuals, and the result is then set back onto the same vector of individuals using the setSpatialPosition() method; however, this method might be useful in other situations too.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Subpopulation: P, addCloned(), addCrossed(), addEmpty(), addRecombinant(), addSelfed(), addSpatialMap(), cachedFitness(), configureDisplay(), defineSpatialMap(), outputMSSample(), outputSample(), outputVCFSample(), pointInBounds(), pointPeriodic(), pointReflected(), pointStopped(), pointUniform(), removeSpatialMap(), removeSubpopulation(), sampleIndividuals(), setCloningRate(), setMigrationRates(), setSelfingRate(), setSexRatio(), setSpatialBounds(), setSubpopulationSize(), spatialMapColor(), spatialMapImage(), spatialMapValue(), subsetIndividuals(), takeMigrants()

---

pointInBounds *SLiM method pointInBounds*

Description

Documentation for SLiM function `pointInBounds`, which is a method of the SLiM class `Subpopulation`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
pointInBounds(point)
```

Arguments

`point`	An object of type float. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 742.

Returns T if point is inside the spatial boundaries of the subpopulation, F otherwise. For example, for a simulation with "xy" dimensionality, if point contains exactly two values constituting an (x,y) point, the result will be T if and only if ((point[0]>=x0) & (point[0]<=x1) & (point[1]>=y0) & (point[1]<=y1)) given spatial bounds (x0, y0, x1, y1). This method is

useful for implementing absorbing or reprising boundary conditions. This may only be called in simulations for which continuous space has been enabled with initializeSLiMOptions(). The length of point must be an exact multiple of the dimensionality of the simulation; in other words, point may contain values comprising more than one point. In this case, a logical vector will be returned in which each element is T if the corresponding point in point is inside the spatial boundaries of the subpopulation, F otherwise.

Value

An object of type logical.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Subpopulation: P, addCloned(), addCrossed(), addEmpty(), addRecombinant(), addSelfed(), addSpatialMap(), cachedFitness(), configureDisplay(), defineSpatialMap(), outputMSSample(), outputSample(), outputVCFSample(), pointDeviated(), pointPeriodic(), pointReflected(), pointStopped(), pointUniform(), removeSpatialMap(), removeSubpopulation(), sampleIndividuals(), setCloningRate(), setMigrationRates(), setSelfingRate(), setSexRatio(), setSpatialBounds(), setSubpopulationSize(), spatialMapColor(), spatialMapImage(), spatialMapValue(), subsetIndividuals(), takeMigrants()

---

pointPeriodic *SLiM method pointPeriodic*

Description

Documentation for SLiM function pointPeriodic, which is a method of the SLiM class Subpopulation. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
pointPeriodic(point)
```

Arguments

`point`	An object of type float. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 742.

Returns a revised version of point that has been brought inside the periodic spatial boundaries of the subpopulation (as specified by the periodicity parameter of initializeSLiMOptions()) by wrapping around periodic spatial boundaries. In brief, if a coordinate of point lies beyond a periodic spatial boundary, that coordinate is wrapped around the boundary, so that it lies inside the spatial extent by the same magnitude that it previously lay outside, but on the opposite side of the space; in effect, the two edges of the periodic spatial boundary are seamlessly joined. This is done iteratively until all coordinates lie inside the subpopulation's periodic boundaries. Note that non-periodic spatial boundaries are not enforced by this method; they should be enforced using pointReflected(), pointStopped(), or some other means of enforcing boundary constraints (which can be used after pointPeriodic() to bring the remaining coordinates into bounds; coordinates already brought into bounds by pointPeriodic() will be unaffected by those calls). This method is useful for implementing periodic boundary conditions. This may only be called in simulations for which continuous space and at least one periodic spatial dimension have been enabled with initializeSLiMOptions(). The length of point must be an exact multiple of the dimensionality of the simulation; in other words, point may contain values comprising more than one point. In this case, each point will be processed as described above and a new vector containing all of the processed points will be returned.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Subpopulation: P, addCloned(), addCrossed(), addEmpty(), addRecombinant(), addSelfed(), addSpatialMap(), cachedFitness(), configureDisplay(), defineSpatialMap(), outputMSSample(), outputSample(), outputVCFSample(), pointDeviated(), pointInBounds(), pointReflected(), pointStopped(), pointUniform(), removeSpatialMap(), removeSubpopulation(), sampleIndividuals(), setCloningRate(), setMigrationRates(), setSelfingRate(),

`setSexRatio()`, `setSpatialBounds()`, `setSubpopulationSize()`, `spatialMapColor()`, `spatialMapImage()`, `spatialMapValue()`, `subsetIndividuals()`, `takeMigrants()`

`pointReflected` *SLiM method pointReflected*

Description

Documentation for SLiM function `pointReflected`, which is a method of the SLiM class `Subpopulation`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
pointReflected(point)
```

Arguments

`point`	An object of type float. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 742.

Returns a revised version of point that has been brought inside the spatial boundaries of the subpopulation by reflection. In brief, if a coordinate of point lies beyond a spatial boundary, that coordinate is reflected across the boundary, so that it lies inside the boundary by the same magnitude that it previously lay outside the boundary. This is done iteratively until all coordinates lie inside the subpopulation's boundaries. This method is useful for implementing reflecting boundary conditions. This may only be called in simulations for which continuous space has been enabled with initializeSLiMOptions(). The length of point must be an exact multiple of the dimensionality of the simulation; in other words, point may contain values comprising more than one point. In this case, each point will be processed as described above and a new vector containing all of the processed points will be returned.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Subpopulation: P, addCloned(), addCrossed(), addEmpty(), addRecombinant(), addSelfed(), addSpatialMap(), cachedFitness(), configureDisplay(), defineSpatialMap(), outputMSSample(), outputSample(), outputVCFSample(), pointDeviated(), pointInBounds(), pointPeriodic(), pointStopped(), pointUniform(), removeSpatialMap(), removeSubpopulation(), sampleIndividuals(), setCloningRate(), setMigrationRates(), setSelfingRate(), setSexRatio(), setSpatialBounds(), setSubpopulationSize(), spatialMapColor(), spatialMapImage(), spatialMapValue(), subsetIndividuals(), takeMigrants()

---

pointStopped *SLiM method pointStopped*

Description

Documentation for SLiM function pointStopped, which is a method of the SLiM class Subpopulation. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
pointStopped(point)
```

Arguments

point	An object of type float. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 743.

Returns a revised version of point that has been brought inside the spatial boundaries of the subpopulation by clamping. In brief, if a coordinate of point lies beyond a spatial boundary, that coordinate is set to exactly the position of the boundary, so that it lies on the edge of the spatial boundary. This method is useful for implementing stopping boundary conditions. This may only be called in simulations for which continuous space has been enabled with initializeSLiMOptions(). The length of point must be an exact multiple of the dimensionality of the simulation; in other words, point may contain values comprising more than one point. In this case, each point will be processed as described above and a new vector containing all of the processed points will be returned.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Subpopulation: P, addCloned(), addCrossed(), addEmpty(), addRecombinant(), addSelfed(), addSpatialMap(), cachedFitness(), configureDisplay(), defineSpatialMap(), outputMSSample(), outputSample(), outputVCFSample(), pointDeviated(), pointInBounds(), pointPeriodic(), pointReflected(), pointUniform(), removeSpatialMap(), removeSubpopulation(), sampleIndividuals(), setCloningRate(), setMigrationRates(), setSelfingRate(), setSexRatio(), setSpatialBounds(), setSubpopulationSize(), spatialMapColor(), spatialMapImage(), spatialMapValue(), subsetIndividuals(), takeMigrants()

---

pointUniform *SLiM method pointUniform*

Description

Documentation for SLiM function `pointUniform`, which is a method of the SLiM class `Subpopulation`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
pointUniform(n)
```

Arguments

`n`	An object of type integer. Must be of length 1 (a singleton). The default value is `1`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 743.

Returns a new point (or points, for n > 1) generated from uniform draws for each coordinate, within the spatial boundaries of the subpopulation. The returned vector will contain n points, each comprised of a number of coordinates equal to the dimensionality of the simulation, so it will be of total length n*dimensionality. This may only be called in simulations for which continuous space has been enabled with initializeSLiMOptions().

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Subpopulation: P, addCloned(), addCrossed(), addEmpty(), addRecombinant(), addSelfed(), addSpatialMap(), cachedFitness(), configureDisplay(), defineSpatialMap(), outputMSSample(), outputSample(), outputVCFSample(), pointDeviated(), pointInBounds(), pointPeriodic(), pointReflected(), pointStopped(), removeSpatialMap(), removeSubpopulation(), sampleIndividuals(), setCloningRate(), setMigrationRates(), setSelfingRate(), setSexRatio(), setSpatialBounds(), setSubpopulationSize(), spatialMapColor(), spatialMapImage(), spatialMapValue(), subsetIndividuals(), takeMigrants()

---

positionsOfMutationsOfType

SLiM method positionsOfMutationsOfType

Description

Documentation for SLiM function `positionsOfMutationsOfType`, which is a method of the SLiM class `Genome`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
positionsOfMutationsOfType(mutType)
```

Arguments

`mutType`	An object of type integer or MutationType object. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 674.

Returns the positions of mutations that are of the type specified by mutType, out of all of the mutations in the genome. If you need a vector of the matching Mutation objects, rather than just positions, use -mutationsOfType(). This method is provided for speed; it is much faster than the corresponding Eidos code.

Value

An object of type integer.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Genome: G, addMutations(), addNewDrawnMutation(), addNewMutation(), containsMarkerMutation(), containsMutations(), countOfMutationsOfType(), mutationCountsInGenomes(), mutationFrequenciesInGe mutationsOfType(), nucleotides(), outputMS(), outputVCF(), output(), readFromMS(), readFromVCF(), removeMutations(), sumOfMutationsOfType()

---

power *SLiM method power*

Description

Documentation for SLiM function power, which is a method of the SLiM class SpatialMap. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
power(x)
```

Arguments

x	An object of type integer or float or SpatialMap object. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 716.

Raises the spatial map to the power x. One possibility is that x is a singleton integer or float value; in this case, each grid value of the target spatial map is raised to the power x. Another possibility is that x is an integer or float vector/matrix/array of the same dimensions as the target spatial map's grid; in this case, each grid value of the target spatial map is raised to the power of the corresponding value of x. The third possibility is that x is itself a (singleton) spatial map; in this case, each grid value of the target spatial map is raised to power of the corresponding grid value of x (and thus the two spatial maps must match in their spatiality, their spatial bounds, and their grid dimensions). The target spatial map is returned, to allow easy chaining of operations.

Value

An object of type SpatialMap object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other SpatialMap: SM, add(), blend(), changeColors(), changeValues(), divide(), exp(), gridValues(), interpolate(), mapColor(), mapImage(), mapValue(), multiply(), range(), rescale(), sampleImprovedNearbyPoint(), sampleNearbyPoint(), smooth(), subtract()

---

range *SLiM method range*

Description

Documentation for SLiM function range, which is a method of the SLiM class SpatialMap. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
range(void)
```

Arguments

`void`	An object of type . See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 716.

Returns the range of values contained in the spatial map. The result is a float vector of length 2; the first element is the minimum map value, and the second element is the maximum map value.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other SpatialMap: SM, add(), blend(), changeColors(), changeValues(), divide(), exp(), gridValues(), interpolate(), mapColor(), mapImage(), mapValue(), multiply(), power(), rescale(), sampleImprovedNearbyPoint(), sampleNearbyPoint(), smooth(), subtract()

---

readFromMS *SLiM method readFromMS*

Description

Documentation for SLiM function `readFromMS`, which is a method of the SLiM class `Genome`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
readFromMS(filePath, mutationType)
```

Arguments

`filePath`	An object of type string. Must be of length 1 (a singleton). See details for description.
`mutationType`	An object of type integer or MutationType object. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 674.

Read new mutations from the MS format file at filePath and add them to the target genomes. The number of target genomes must match the number of genomes represented in the MS file. To read into all of the genomes in a given subpopulation pN, simply call pN.genomes.readFromMS(), assuming the subpopulation's size matches that of the MS file. A vector containing all of the mutations created by readFromMS() is returned. Each mutation is created at the position specified in the file, using the mutation type given by mutationType. Positions are expected to be in [0,1], and are scaled to the length of the chromosome by multiplying by the last valid base position of the chromosome (i.e., one less than the chromosome length). Selection coefficients are drawn from the mutation type. The population of origin for each mutation is set to -1, and the tick of origin is set to the current tick. In a nucleotide-based model, if mutationType is nucleotide-based, a random nucleotide different from the ancestral nucleotide at the position will be chosen with equal probability. The target genomes correspond, in order, to the call lines in the MS file. In sex-based models that simulate the X or Y chromosome, null genomes in the target vector will be skipped, and will not be used to correspond to any call line; however, care should be taken in this case that the lines in the MS file correspond to the target genomes in the manner desired.

Value

An object of type Mutation object.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Genome: `G`, `addMutations()`, `addNewDrawnMutation()`, `addNewMutation()`, `containsMarkerMutation()`, `containsMutations()`, `countOfMutationsOfType()`, `mutationCountsInGenomes()`, `mutationFrequenciesInGe`
`mutationsOfType()`, `nucleotides()`, `outputMS()`, `outputVCF()`, `output()`, `positionsOfMutationsOfType()`, `readFromVCF()`, `removeMutations()`, `sumOfMutationsOfType()`

`readFromPopulationFile`

SLiM method readFromPopulationFile

Description

Documentation for SLiM function `readFromPopulationFile`, which is a method of the SLiM class `Species`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
readFromPopulationFile(filePath, subpopMap)
```

Arguments

`filePath`	An object of type string. Must be of length 1 (a singleton). See details for description.
`subpopMap`	An object of type null. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 723.

Read from a population initialization file, whether in text or binary format as previously specified to outputFull(), and return the tick counter value represented by the file's contents (i.e., the tick at which the file was generated). Although this is most commonly used to set up initial populations (often in an Eidos event set to run in tick 1, immediately after simulation initialization), it may be called in any early() or late() Eidos event; the current state of all populations in the target species will be wiped and replaced by the state in the file at filePath. All Eidos variables that are of type object and have element type Subpopulation, Genome, Mutation, Individual, or Substitution will be removed as a side effect of this method if they contain any element that belongs to the target species, because those objects will no longer exist in the SLiM simulation; if you want to preserve any of that state, you should output it or save it to a file prior to this call. New symbols will be defined to refer to the new Subpopulation objects loaded from the file. If the file being read was written by a version of SLiM prior to 2.3, then for backward compatibility fitness values will be calculated immediately for any new subpopulations created by this call, which will trigger

the calling of any activated and applicable mutationEffect() and fitnessEffect() callbacks. When reading files written by SLiM 2.3 or later, fitness values are not calculated as a side effect of this call (because the simulation will often need to evaluate interactions or modify other state prior to doing so). In SLiM 2.3 and later when using the WF model, calling readFromPopulationFile() from any context other than a late() event causes a warning; calling from a late() event is almost always correct in WF models, so that fitness values can be automatically recalculated by SLiM at the usual time in the tick cycle without the need to force their recalculation (see chapter 23, and comments on recalculateFitness() below). In SLiM 3.0 when using the nonWF model, calling readFromPopulationFile() from any context other than an early() event causes a warning; calling from an early() event is almost always correct in nonWF models, so that fitness values can be automatically recalculated by SLiM at the usual time in the tick cycle without the need to force their recalculation (see chapter 24, and comments on recalculateFitness() below). As of SLiM 2.1, this method changes the tick and cycle counters to the tick and cycle read from the file. If you do not want these counters to be changed, you can change them back after reading, by setting community.tick and sim.cycle to whatever values you wish. Note that restoring a saved past state and running forward again will not yield the same simulation results, because the random number generator's state will not be the same; to ensure reproducibility from a given time point, setSeed() can be used to establish a new seed value. Any changes made to structure of the species (mutation types, genomic element types, etc.) will not be wiped and re-established by readFromPopulationFile(); this method loads only the population's state, not the species configuration, so care should be taken to ensure that the species structure meshes coherently with the loaded data. Indeed, state such as the selfing and cloning rates of subpopulations, values set into tag properties, and values set onto objects with setValue() will also be lost, since it is not saved out by outputFull(). Only information saved by outputFull() will be restored; all other state associated with the species - subpopulations, individuals, genomes, mutations, and substitutions - will be lost, and should be re-established by the model if it is still needed. As of SLiM 2.3, this method will read and restore the spatial positions of individuals if that information is present in the output file and the species has enabled continuous space (see outputFull() for details). If spatial positions are present in the output file but the species has not enabled continuous space (or the number of spatial dimensions does not match), an error will result. If the species has enabled continuous space but spatial positions are not present in the output file, the spatial positions of the individuals read will be undefined, but an error is not raised. As of SLiM 3.0, this method will read and restore the ages of individuals if that information is present in the output file and the simulation is based upon the nonWF model. If ages are present but the simulation uses a WF model, an error will result; the WF model does not use age information. If ages are not present but the simulation uses a nonWF model, an error will also result; the nonWF model requires age information. As of SLiM 3.3, this method will restore the nucleotides of nucleotide-based mutations, and will restore the ancestral nucleotide sequence, if that information is present in the output file. Loading an output file that contains nucleotide information in a non-nucleotide-based model, and vice versa, will produce an error. As of SLiM 3.5, this method will read and restore the pedigree IDs of individuals and genomes if that information is present in the output file (as requested with outputFull(pedigreeIDs=T)) and if SLiM's optional pedigree tracking has been enabled with initializeSLiMOptions(keepPedigrees=T). This method can also be used to read tree-sequence (.trees) files saved by treeSeqOutput() or generated by the Python pyslim package. Note that the user metadata for a tree-sequence file can be read separately

with the treeSeqMetadata() function. Beginning with SLiM 4, the subpopMap parameter may be supplied to re-order the populations of the input tree sequence when it is loaded in to SLiM. This parameter must have a value that is a Dictionary; the keys of this dictionary should be SLiM population identifiers as string values (e.g., "p2"), and the values should be indexes of populations in the input tree sequence; a key/value pair of "p2", 4 would mean that the fifth population in the input (the one at zero-based index 4) should become p2 on loading into SLiM. If subpopMap is non-NULL, all populations in the tree sequence must be explicitly mapped, even if their index will not change and even if they will not be used by SLiM; the only exception is for unused slots in the population table, which can be explicitly remapped but do not have to be. For instance, suppose we have a tree sequence in which population 0 is unused, population 1 is not a SLiM population (for example, an ancestral population produced by msprime), and population 2 is a SLiM population, and we want to load this in with population 2 as p0 in SLiM. To do this, we could supply a value of Dictionary("p0", 2, "p1", 1, "p2", 0) for subpopMap, or we could leave out slot 0 since it is unused, with Dictionary("p0", 2, "p1", 1). Although this facility cannot be used to remove populations in the tree sequence, note that it may add populations that will be visible when treeSeqOutput() is called (although these will not be SLiM populations); if, in this example, we had used Dictionary("p0", 0, "p1", 1, "p5", 2) and then we wrote the result out with treeSeqOutput(), the resulting tree sequence would have six populations, although three of them would be empty and would not be used by SLiM. The use of subpopMap makes it easier to load simulation data that was generated in Python, since that typically uses an id of 0. The subpopMap parameter may not be used with file formats other than tree-sequence files, at the present time; setting up the correct subpopulation ids is typically easier when working with those other formats. Note the tskit command-line interface can be used, like python3 -m tskit populations file.trees, to find out the number of subpopulations in a tree-sequence file and their IDs. When loading a tree sequence, a crosscheck of the loaded data will be performed to ensure that the tree sequence was well-formed and was loaded correctly. When running a Release build of SLiM, however, this crosscheck will only occur the first time that readFromPopulationFile() is called to load a tree sequence; subsequent calls will not perform this crosscheck, for greater speed when running models that load saved population state many times (such as models that are conditional on fixation). If you suspect that a tree sequence file might be corrupted or read incorrectly, running a Debug build of SLiM enables crosschecks after every load.

Value

An object of type integer. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Species: Sp, addSubpopSplit(), addSubpop(), countOfMutationsOfType(), individualsWithPedigreeID
killIndividuals(), mutationCounts(), mutationFrequencies(), mutationsOfType(),
outputFixedMutations(), outputFull(), outputMutations(), recalculateFitness(),
registerFitnessEffectCallback(), registerMateChoiceCallback(), registerModifyChildCallback(),
registerMutationCallback(), registerMutationEffectCallback(), registerRecombinationCallback(),
registerReproductionCallback(), registerSurvivalCallback(), simulationFinished(),
skipTick(), subsetMutations(), treeSeqCoalesced(), treeSeqOutput(), treeSeqRememberIndividuals(),
treeSeqSimplify()

---

readFromVCF *SLiM method readFromVCF*

Description

Documentation for SLiM function `readFromVCF`, which is a method of the SLiM class Genome. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
readFromVCF(filePath, mutationType)
```

Arguments

`filePath`	An object of type string. Must be of length 1 (a singleton). See details for description.
`mutationType`	An object of type null or integer or MutationType object. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 675.

Read new mutations from the VCF format file at filePath and add them to the target genomes. The number of target genomes must match the number of genomes represented in the VCF file (i.e., two times the number of samples, if each sample is diploid). To read into all of the genomes in a given subpopulation pN, simply call pN.genomes.readFromVCF(), assuming the subpopulation's size matches that of the VCF file taking ploidy into account. A vector containing all of the mutations created by readFromVCF() is returned. SLiM's VCF parsing is quite primitive. The header is parsed only inasmuch as SLiM looks to see whether SLiM-specific VCF fields (see sections 27.2.3 and 27.2.4) are defined or not; the rest of the header information is ignored. Call lines are assumed to follow the format: #CHROM POS ID REF ALT QUAL FILTER INFO FORMAT i0...iN The CHROM, ID, QUAL, FILTER, and FORMAT fields are ignored, and information in the genotype fields

beyond the GT genotype subfield are also ignored. SLiM's own VCF annotations (see section 27.2.3) are honored; in particular, mutations will be created using the given values of MID, S, PO, TO, and MT if those subfields are present, and DOM, if it is present, must match the dominance coefficient of the mutation type. The parameter mutationType (a MutationType object or id) will be used for any mutations that have no supplied mutation type id in the MT subfield; if mutationType would be used but is NULL an error will result. Mutation IDs supplied in MID will be used if no mutation IDs have been used in the simulation so far; if any have been used, it is difficult for SLiM to guarantee that there are no conflicts, so a warning will be emitted and the MID values will be ignored. If selection coefficients are not supplied with the S subfield, they will be drawn from the mutation type used for the mutation. If a population of origin is not supplied with the PO subfield, -1 will be used. If a tick of origin is not supplied with the TO subfield (or a generation of origin GO field, which was the SLiM convention before SLiM 4), the current tick will be used. REF and ALT must always be comprised of simple nucleotides (A/C/G/T) rather than values representing indels or other complex states. Beyond this, the handling of the REF and ALT fields depends upon several factors. First of all, these fields are ignored in non-nucleotide-based models, although they are still checked for conformance. In nucleotide-based models, when a header definition for SLiM's NONNUC tag is present (as when nucleotide-based output is generated by SLiM): Second, if a NONNUC field is present in the INFO field the call line is taken to represent a non-nucleotide-based mutation, and REF and ALT are again ignored. In this case the mutation type used must be non-nucleotidebased. Third, if NONNUC is not present the call line is taken to represent a nucleotide-based mutation. In this case, the mutation type used must be nucleotide-based. Also, in this case, the specified reference nucleotide must match the existing ancestral nucleotide at the given position. In nucleotidebased models, when a header definition for SLiM's NONNUC tag is not present (as when loading a non- SLiM-generated VCF file): The mutation type will govern the way nucleotides are handled. If the mutation type used for a mutation is nucleotide-based, the nucleotide provided in the VCF file for that allele will be used. If the mutation type is non-nucleotide-based, the nucleotide provided will be ignored. If multiple alleles using the same nucleotide at the same position are specified in the VCF file, a separate mutation will be created for each, mirroring SLiM's behavior with independent mutational lineages when writing VCF (see section 27.2.4). The MULTIALLELIC flag is ignored by readFromVCF(); call lines for mutations at the same base position in the same genome will result in stacked mutations whether or not MULTIALLELIC is present. The target genomes correspond, in order, to the haploid or diploid calls provided for i0…iN (the sample IDs) in the VCF file. In sex-based models that simulate the X or Y chromosome, null genomes in the target vector will be skipped, and will not be used to correspond to any of i0…iN; however, care should be taken in this case that the genomes in the VCF file correspond to the target genomes in the manner desired.

Value

An object of type Mutation object.

Copyright

More information about SLiM can be found on the official website: https://messerlab.org/slim/

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Genome: G, addMutations(), addNewDrawnMutation(), addNewMutation(), containsMarkerMutation(), containsMutations(), countOfMutationsOfType(), mutationCountsInGenomes(), mutationFrequenciesInGe mutationsOfType(), nucleotides(), outputMS(), outputVCF(), output(), positionsOfMutationsOfType(), readFromMS(), removeMutations(), sumOfMutationsOfType()

---

recalculateFitness *SLiM method recalculateFitness*

Description

Documentation for SLiM function recalculateFitness, which is a method of the SLiM class Species. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
recalculateFitness(tick)
```

Arguments

tick	An object of type null or integer. Must be of length 1 (a singleton). The default value is NULL. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 725.

Force an immediate recalculation of fitness values for all individuals in all subpopulations. Normally fitness values are calculated at a fixed point in each tick, and those values are cached and used until the next recalculation. If simulation parameters are changed in script in a way that affects fitness calculations, and if you wish those changes to take effect immediately rather than taking effect at the next automatic recalculation, you may call recalculateFitness() to force an immediate recalculation and recache. The optional parameter tick provides the tick for which mutationEffect() and fitnessEffect() callbacks should be selected; if it is NULL (the default), the current tick value for the simulation, community.tick, is used. If you call recalculateFitness() in an early() event in a WF model, you may wish this to be community.tick - 1 in order to utilize the mutationEffect() and fitnessEffect()

callbacks for the previous tick, as if the changes that you have made to fitnessinfluencing parameters were already in effect at the end of the previous tick when the new generation was first created and evaluated (usually it is simpler to just make such changes in a late() event instead, however, in which case calling recalculateFitness() is probably not necessary at all since fitness values will be recalculated immediately afterwards). Regardless of the value supplied for tick here, community.tick inside callbacks will report the true tick number, so if your callbacks consult that parameter in order to create tick-specific fitness effects you will need to handle the discrepancy somehow. (Similar considerations apply for nonWF models that call recalculateFitness() in a late() event, which is also not advisable in general.) After this call, the fitness values used for all purposes in SLiM will be the newly calculated values. Calling this method will trigger the calling of any enabled and applicable mutationEffect() and fitnessEffect() callbacks, so this is quite a heavyweight operation; you should think carefully about what side effects might result (which is why fitness recalculation does not just occur automatically after changes that might affect fitness values).

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Species: Sp, addSubpopSplit(), addSubpop(), countOfMutationsOfType(), individualsWithPedigreeID
killIndividuals(), mutationCounts(), mutationFrequencies(), mutationsOfType(),
outputFixedMutations(), outputFull(), outputMutations(), readFromPopulationFile(),
registerFitnessEffectCallback(), registerMateChoiceCallback(), registerModifyChildCallback(),
registerMutationCallback(), registerMutationEffectCallback(), registerRecombinationCallback(),
registerReproductionCallback(), registerSurvivalCallback(), simulationFinished(),
skipTick(), subsetMutations(), treeSeqCoalesced(), treeSeqOutput(), treeSeqRememberIndividuals(),
treeSeqSimplify()

---

recombination *SLiM recombination() callback*

Description

This callback specifies that a code block is providing logic to determine the breakpoints in the genome for recombination. It is called during the generation of gametes for each gamete in the generations it is active. This is achieved by setting breakpoints into the pseudo-variable `breakpoints`, replacing the default (as integer positions). If `breakpoints` is modified the callback must return `T`. If `return(F)` is used, SLiM will use the original `breakpoints`. see SLiM Manual: page 604

Usage

```
recombination(subpop_id)
```

Arguments

`subpop_id`	The id(s) of the subpopulation(s) to which this callback should apply. Can be an integer 1, 2, etc., or character "p1", "p2", etc.

Details

Global variables available in reproduction callbacks:

individual The focal parent that is generating a gamete

genome1 One genome of the focal parent; this is the initial copy strand

genome2 The other genome of the focal parent

subpop The subpopulation to which the focal parent belongs

breakpoints An integer vector of crossover breakpoints

Value

None

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other callbacks: `early()`, `first()`, `fitnessEffect()`, `fitness()`, `initialize()`, `interaction()`, `late()`, `mateChoice()`, `modifyChild()`, `mutationEffect()`, `mutation()`, `reproduction()`, `slim_callbacks()`, `survival()`

Examples

```
slim_block(recombination(), {
  if (genome1.containsMarkerMutation(m2, 25000) ==
      genome2.containsMarkerMutation(m2, 25000)) {

    return(F)

  }

  inInv = (breakpoints > 25000) & (breakpoints < 75000)
  if (!any(inInv)) {
    return(F)
  }

  breakpoints = breakpoints[!inInv]
  return(T)

})
```

`reconstruct` *Reconstruct slimrlang code to make this slimr_script*

Description

This reconstructs a `slimrlang` input sequence to regenerate the given `slimr_script` object. This is useful if you want to edit the SLiM script to add additional functionality, for example, where you want to incorporate the results of `slimrlang`'s internal edits, e.g. such as removing `%.%` special operators, etc. It is also useful when the `slimr_script` object has been created from converting a text-based SLiM script, such as when using `as.slimr_script` from the `slimr` package on a character variable.

Usage

```
reconstruct(x, ...)
```

Arguments

`x`	slimr_script object to reconstruct
`...`	Further arguments, passed to or from other methods.

Value

A character vector of length one containing the reconstructed code.

Examples

```
slim_script(
    slim_block(initialize(),
         {
              .Init$initializeMutationRate(1e-7);
              .Init$initializeMutationType("m1", 0.5, "f", 0.0);
              .Init$initializeGenomicElementType("g1", m1, 1.0);
              .Init$initializeGenomicElement(g1, 0, 99999);
              .Init$initializeRecombinationRate(1e-8);
         }),
           slim_block(1,
           {
              sim%.%.SS$addSubpop("p1", 500);
           }),
   slim_block(10000,
         {
              sim%.%.SS$simulationFinished();
         })
) -> script
reconstruct(script)
```

---

reconstruct.slimr_script

Reconstruct slimrlang code to make this slimr_script

Description

This reconstructs a `slimrlang` input sequence to regenerate the given `slimr_script` object. This is useful if you want to edit the SLiM script to add additional functionality, for example, where you want to incorporate the results of `slimrlang`'s internal edits, e.g. such as removing `%.%` special operators, etc. It is also useful when the `slimr_script` object has been created from converting a text-based SLiM script, such as when using `as.slimr_script` from the `slimr` package on a character variable.

Usage

```
## S3 method for class 'slimr_script'
reconstruct(x, ...)
```

Arguments

`x`	slimr_script object to reconstruct
`...`	Further arguments, passed to or from other methods.

Value

A character vector of length one containing the reconstructed code.

Examples

```
slim_script(
    slim_block(initialize(),
         {
              .Init$initializeMutationRate(1e-7);
              .Init$initializeMutationType("m1", 0.5, "f", 0.0);
              .Init$initializeGenomicElementType("g1", m1, 1.0);
              .Init$initializeGenomicElement(g1, 0, 99999);
              .Init$initializeRecombinationRate(1e-8);
         }),
           slim_block(1,
           {
              sim%.%.SS$addSubpop("p1", 500);
           }),
   slim_block(10000,
         {
              sim%.%.SS$simulationFinished();
         })
) -> script
reconstruct(script)
```

---

registerEarlyEvent *SLiM method registerEarlyEvent*

Description

Documentation for SLiM function `registerEarlyEvent`, which is a method of the SLiM class `Community`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
registerEarlyEvent(id, source, start, end, ticksSpec)
```

Arguments

`id`	An object of type null or integer or string. Must be of length 1 (a singleton). See details for description.
`source`	An object of type string. Must be of length 1 (a singleton). See details for description.
`start`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`end`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`ticksSpec`	An object of type null or Species object. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 666.

Register a block of Eidos source code, represented as the string singleton source, as an Eidos early() event in the current simulation, with optional start and end ticks (and, for multispecies models, optional ticks specifier ticksSpec) limiting its applicability. The script block will be given identifier id (specified as an integer, or as a string symbolic name such as "s5"); this may be NULL if there is no need to be able to refer to the block later. The registered event is added to the end of the list of registered SLiMEidosBlock objects, and is active immediately; it may be eligible to execute in the current tick (see section 26.11 for details). The new SLiMEidosBlock will be defined as a global variable immediately by this method (see section 25.12), and will also be returned by this method.

Value

An object of type SLiMEidosBlock object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Community: Co, createLogFile(), deregisterScriptBlock(), genomicElementTypesWithIDs(), interactionTypesWithIDs(), mutationTypesWithIDs(), outputUsage(), registerFirstEvent(), registerInteractionCallback(), registerLateEvent(), rescheduleScriptBlock(), scriptBlocksWithIDs(), simulationFinished(), speciesWithIDs(), subpopulationsWithIDs(), usage()

---

registerFirstEvent *SLiM method registerFirstEvent*

Description

Documentation for SLiM function registerFirstEvent, which is a method of the SLiM class Community. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
registerFirstEvent(id, source, start, end, ticksSpec)
```

Arguments

`id`	An object of type null or integer or string. Must be of length 1 (a singleton). See details for description.
`source`	An object of type string. Must be of length 1 (a singleton). See details for description.
`start`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`end`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`ticksSpec`	An object of type null or Species object. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 667.

Register a block of Eidos source code, represented as the string singleton source, as an Eidos first() event in the current simulation, with optional start and end ticks (and, for multispecies models, optional ticks specifier ticksSpec) limiting its applicability. The script block will be given identifier id (specified as an integer, or as a string symbolic name such as "s5"); this may be NULL if there is no need to be able to refer to the block later. The registered event is added to the end of the list of registered SLiMEidosBlock objects, and is active immediately; it may be eligible to execute in the current tick (see section 26.11 for details). The new SLiMEidosBlock will be defined as a global variable immediately by this method (see section 25.12), and will also be returned by this method.

Value

An object of type SLiMEidosBlock object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Community: Co, createLogFile(), deregisterScriptBlock(), genomicElementTypesWithIDs(), interactionTypesWithIDs(), mutationTypesWithIDs(), outputUsage(), registerEarlyEvent(), registerInteractionCallback(), registerLateEvent(), rescheduleScriptBlock(), scriptBlocksWithIDs(), simulationFinished(), speciesWithIDs(), subpopulationsWithIDs(), usage()

---

registerFitnessEffectCallback

SLiM method registerFitnessEffectCallback

Description

Documentation for SLiM function `registerFitnessEffectCallback`, which is a method of the SLiM class Species. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
registerFitnessEffectCallback(id, source, subpop, start, end)
```

Arguments

`id`	An object of type null or integer or string. Must be of length 1 (a singleton). See details for description.
`source`	An object of type string. Must be of length 1 (a singleton). See details for description.
`subpop`	An object of type null or integer or Subpopulation object. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`start`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`end`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 725.

Register a block of Eidos source code, represented as the string singleton source, as an Eidos fitnessEffect() callback in the current simulation (specific to the target species), with an optional subpopulation subpop (which may be an integer identifier, or NULL, the default, to indicate all subpopulations), and optional start and end ticks all limiting its applicability. The script block will be given identifier id (specified as an integer, or as a string symbolic name such as "s5"); this may be NULL if there is no need to be able to refer to the block

later. The registered callback is added to the end of the list of registered SLiMEidosBlock objects, and is active immediately; it may be eligible to execute in the current tick (see section 26.11 for details). The new SLiMEidosBlock will be defined as a global variable immediately by this method (see section 25.12), and will also be returned by this method.

Value

An object of type SLiMEidosBlock object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Species: Sp, addSubpopSplit(), addSubpop(), countOfMutationsOfType(), individualsWithPedigreeID killIndividuals(), mutationCounts(), mutationFrequencies(), mutationsOfType(), outputFixedMutations(), outputFull(), outputMutations(), readFromPopulationFile(), recalculateFitness(), registerMateChoiceCallback(), registerModifyChildCallback(), registerMutationCallback(), registerMutationEffectCallback(), registerRecombinationCallback(), registerReproductionCallback(), registerSurvivalCallback(), simulationFinished(), skipTick(), subsetMutations(), treeSeqCoalesced(), treeSeqOutput(), treeSeqRememberIndividuals(), treeSeqSimplify()

---

registerInteractionCallback

SLiM method registerInteractionCallback

Description

Documentation for SLiM function `registerInteractionCallback`, which is a method of the SLiM class `Community`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
registerInteractionCallback(id, source, intType, subpop, start, end)
```

Arguments

`id`	An object of type null or integer or string. Must be of length 1 (a singleton). See details for description.
`source`	An object of type string. Must be of length 1 (a singleton). See details for description.
`intType`	An object of type integer or InteractionType object. Must be of length 1 (a singleton). See details for description.
`subpop`	An object of type null or integer or Subpopulation object. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`start`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`end`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 667.

Register a block of Eidos source code, represented as the string singleton source, as an Eidos interaction() callback in the current simulation (global to the community), with a required interaction type intType (which may be an integer identifier), optional exerter subpopulation subpop (which may also be an integer identifier, or NULL, the default, to indicate all subpopulations), and optional start and end ticks all limiting its applicability. The script block will be given identifier id (specified as an integer, or as a string symbolic name such as "s5"); this may be NULL if there is no need to be able to refer to the block later. The registered callback is added to the end of the list of registered SLiMEidosBlock objects, and is active immediately; it will be eligible to execute the next time an InteractionType is evaluated. The new SLiMEidosBlock will be defined as a global variable immediately by this method (see section 25.12), and will also be returned by this method.

Value

An object of type SLiMEidosBlock object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Community: `Co`, `createLogFile()`, `deregisterScriptBlock()`, `genomicElementTypesWithIDs()`, `interactionTypesWithIDs()`, `mutationTypesWithIDs()`, `outputUsage()`, `registerEarlyEvent()`, `registerFirstEvent()`, `registerLateEvent()`, `rescheduleScriptBlock()`, `scriptBlocksWithIDs()`, `simulationFinished()`, `speciesWithIDs()`, `subpopulationsWithIDs()`, `usage()`

`registerLateEvent` *SLiM method registerLateEvent*

Description

Documentation for SLiM function `registerLateEvent`, which is a method of the SLiM class `Community`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
registerLateEvent(id, source, start, end, ticksSpec)
```

Arguments

`id`	An object of type null or integer or string. Must be of length 1 (a singleton). See details for description.
`source`	An object of type string. Must be of length 1 (a singleton). See details for description.
`start`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`end`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`ticksSpec`	An object of type null or Species object. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 667.

Register a block of Eidos source code, represented as the string singleton source, as an Eidos late() event in the current simulation, with optional start and end ticks (and, for multispecies models, optional ticks specifier ticksSpec) limiting its applicability. The script block will be given identifier id (specified as an integer, or as a string symbolic name such as "s5"); this may be NULL if there is no need to be able to refer to the block later. The registered event is added to the end of the list of registered SLiMEidosBlock objects, and is active immediately; it may be eligible to execute in the current tick (see section 26.11 for details). The new SLiMEidosBlock will be defined as a global variable immediately by this method (see section 25.12), and will also be returned by this method.

Value

An object of type SLiMEidosBlock object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Community: Co, createLogFile(), deregisterScriptBlock(), genomicElementTypesWithIDs(), interactionTypesWithIDs(), mutationTypesWithIDs(), outputUsage(), registerEarlyEvent(), registerFirstEvent(), registerInteractionCallback(), rescheduleScriptBlock(), scriptBlocksWithIDs(), simulationFinished(), speciesWithIDs(), subpopulationsWithIDs(), usage()

---

registerMateChoiceCallback

SLiM method registerMateChoiceCallback

Description

Documentation for SLiM function `registerMateChoiceCallback`, which is a method of the SLiM class `Species`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
registerMateChoiceCallback(id, source, subpop, start, end)
```

Arguments

`id`	An object of type null or integer or string. Must be of length 1 (a singleton). See details for description.
`source`	An object of type string. Must be of length 1 (a singleton). See details for description.
`subpop`	An object of type null or integer or Subpopulation object. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

`start` An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

`end` An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 725.

Register a block of Eidos source code, represented as the string singleton source, as an Eidos mateChoice() callback in the current simulation (specific to the target species), with optional subpopulation subpop (which may be an integer identifier, or NULL, the default, to indicate all subpopulations) and optional start and end ticks all limiting its applicability. The script block will be given identifier id (specified as an integer, or as a string symbolic name such as "s5"); this may be NULL if there is no need to be able to refer to the block later. The registered callback is added to the end of the list of registered SLiMEidosBlock objects, and is active immediately; it may be eligible to execute in the current tick (see section 26.11 for details). The new SLiMEidosBlock will be defined as a global variable immediately by this method (see section 25.12), and will also be returned by this method.

Value

An object of type SLiMEidosBlock object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Species: Sp, addSubpopSplit(), addSubpop(), countOfMutationsOfType(), individualsWithPedigreeID killIndividuals(), mutationCounts(), mutationFrequencies(), mutationsOfType(), outputFixedMutations(), outputFull(), outputMutations(), readFromPopulationFile(), recalculateFitness(), registerFitnessEffectCallback(), registerModifyChildCallback(), registerMutationCallback(), registerMutationEffectCallback(), registerRecombinationCallback(), registerReproductionCallback(), registerSurvivalCallback(), simulationFinished(), skipTick(), subsetMutations(), treeSeqCoalesced(), treeSeqOutput(), treeSeqRememberIndividuals(), treeSeqSimplify()

`registerModifyChildCallback`
SLiM method registerModifyChildCallback

Description

Documentation for SLiM function `registerModifyChildCallback`, which is a method of the SLiM class `Species`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
registerModifyChildCallback(id, source, subpop, start, end)
```

Arguments

`id`	An object of type null or integer or string. Must be of length 1 (a singleton). See details for description.
`source`	An object of type string. Must be of length 1 (a singleton). See details for description.
`subpop`	An object of type null or integer or Subpopulation object. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`start`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`end`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 726.

Register a block of Eidos source code, represented as the string singleton source, as an Eidos modifyChild() callback in the current simulation (specific to the target species), with optional subpopulation subpop (which may be an integer identifier, or NULL, the default, to indicate all subpopulations) and optional start and end ticks all limiting its applicability. The script block will be given identifier id (specified as an integer, or as a string symbolic name such as "s5"); this may be NULL if there is no need to be able to refer to the block later. The registered callback is added to the end of the list of registered SLiMEidosBlock objects, and is active immediately; it may be eligible to execute in the current tick (see section 26.11 for details). The new SLiMEidosBlock will be defined as a global variable immediately by this method (see section 25.12), and will also be returned by this method.

Value

An object of type SLiMEidosBlock object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Species: Sp, addSubpopSplit(), addSubpop(), countOfMutationsOfType(), individualsWithPedigreeID killIndividuals(), mutationCounts(), mutationFrequencies(), mutationsOfType(), outputFixedMutations(), outputFull(), outputMutations(), readFromPopulationFile(), recalculateFitness(), registerFitnessEffectCallback(), registerMateChoiceCallback(), registerMutationCallback(), registerMutationEffectCallback(), registerRecombinationCallback(), registerReproductionCallback(), registerSurvivalCallback(), simulationFinished(), skipTick(), subsetMutations(), treeSeqCoalesced(), treeSeqOutput(), treeSeqRememberIndividuals(), treeSeqSimplify()

---

registerMutationCallback

SLiM method registerMutationCallback

Description

Documentation for SLiM function `registerMutationCallback`, which is a method of the SLiM class Species. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
registerMutationCallback(id, source, mutType, subpop, start, end)
```

Arguments

id	An object of type null or integer or string. Must be of length 1 (a singleton). See details for description.
source	An object of type string. Must be of length 1 (a singleton). See details for description.
mutType	An object of type null or integer or MutationType object. Must be of length 1 (a singleton). The default value is NULL. See details for description.

`subpop`	An object of type null or integer or Subpopulation object. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`start`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`end`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 726.

Register a block of Eidos source code, represented as the string singleton source, as an Eidos mutation() callback in the current simulation (specific to the target species), with an optional mutation type mutType (which may be an integer mutation type identifier, or NULL, the default, to indicate all mutation types - see section 26.9), optional subpopulation subpop (which may also be an integer identifier, or NULL, the default, to indicate all subpopulations), and optional start and end ticks all limiting its applicability. The script block will be given identifier id (specified as an integer, or as a string symbolic name such as "s5"); this may be NULL if there is no need to be able to refer to the block later. The registered callback is added to the end of the list of registered SLiMEidosBlock objects, and is active immediately; it may be eligible to execute in the current tick (see section 26.11 for details). The new SLiMEidosBlock will be defined as a global variable immediately by this method (see section 25.12), and will also be returned by this method.

Value

An object of type SLiMEidosBlock object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Species: Sp, addSubpopSplit(), addSubpop(), countOfMutationsOfType(), individualsWithPedigreeID
killIndividuals(), mutationCounts(), mutationFrequencies(), mutationsOfType(),
outputFixedMutations(), outputFull(), outputMutations(), readFromPopulationFile(),
recalculateFitness(), registerFitnessEffectCallback(), registerMateChoiceCallback(),
registerModifyChildCallback(), registerMutationEffectCallback(), registerRecombinationCallback(),
registerReproductionCallback(), registerSurvivalCallback(), simulationFinished(),

`skipTick()`, `subsetMutations()`, `treeSeqCoalesced()`, `treeSeqOutput()`, `treeSeqRememberIndividuals()`, `treeSeqSimplify()`

`registerMutationEffectCallback`
SLiM method registerMutationEffectCallback

Description

Documentation for SLiM function `registerMutationEffectCallback`, which is a method of the SLiM class `Species`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
registerMutationEffectCallback(id, source, mutType, subpop, start, end)
```

Arguments

`id`	An object of type null or integer or string. Must be of length 1 (a singleton). See details for description.
`source`	An object of type string. Must be of length 1 (a singleton). See details for description.
`mutType`	An object of type integer or MutationType object. Must be of length 1 (a singleton). See details for description.
`subpop`	An object of type null or integer or Subpopulation object. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`start`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`end`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 726.

Register a block of Eidos source code, represented as the string singleton source, as an Eidos mutationEffect() callback in the current simulation (specific to the target species), with a required mutation type mutType (which may be an integer mutation type identifier), optional subpopulation subpop (which may also be an integer identifier, or NULL, the default, to indicate all subpopulations), and optional start and end ticks all limiting its applicability. The script block will be given identifier id (specified as an integer, or as a string symbolic name such as "s5"); this may be NULL if there is no need to be able to refer to the block later. The registered callback is added to the end of the list of registered

SLiMEidosBlock objects, and is active immediately; it may be eligible to execute in the current tick (see section 26.11 for details). The new SLiMEidosBlock will be defined as a global variable immediately by this method (see section 25.12), and will also be returned by this method.

Value

An object of type SLiMEidosBlock object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Species: Sp, addSubpopSplit(), addSubpop(), countOfMutationsOfType(), individualsWithPedigreeID
killIndividuals(), mutationCounts(), mutationFrequencies(), mutationsOfType(),
outputFixedMutations(), outputFull(), outputMutations(), readFromPopulationFile(),
recalculateFitness(), registerFitnessEffectCallback(), registerMateChoiceCallback(),
registerModifyChildCallback(), registerMutationCallback(), registerRecombinationCallback(),
registerReproductionCallback(), registerSurvivalCallback(), simulationFinished(),
skipTick(), subsetMutations(), treeSeqCoalesced(), treeSeqOutput(), treeSeqRememberIndividuals(),
treeSeqSimplify()

---

registerRecombinationCallback

SLiM method registerRecombinationCallback

Description

Documentation for SLiM function registerRecombinationCallback, which is a method of the SLiM class Species. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
registerRecombinationCallback(id, source, subpop, start, end)
```

Arguments

`id`	An object of type null or integer or string. Must be of length 1 (a singleton). See details for description.
`source`	An object of type string. Must be of length 1 (a singleton). See details for description.
`subpop`	An object of type null or integer or Subpopulation object. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`start`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`end`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 726.

Register a block of Eidos source code, represented as the string singleton source, as an Eidos recombination() callback in the current simulation (specific to the target species), with optional subpopulation subpop (which may be an integer identifier, or NULL, the default, to indicate all subpopulations) and optional start and end ticks all limiting its applicability. The script block will be given identifier id (specified as an integer, or as a string symbolic name such as "s5"); this may be NULL if there is no need to be able to refer to the block later. The registered callback is added to the end of the list of registered SLiMEidosBlock objects, and is active immediately; it may be eligible to execute in the current tick (see section 26.11 for details). The new SLiMEidosBlock will be defined as a global variable immediately by this method (see section 25.12), and will also be returned by this method.

Value

An object of type SLiMEidosBlock object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Species: Sp, addSubpopSplit(), addSubpop(), countOfMutationsOfType(), individualsWithPedigreeID
killIndividuals(), mutationCounts(), mutationFrequencies(), mutationsOfType(),
outputFixedMutations(), outputFull(), outputMutations(), readFromPopulationFile(),
recalculateFitness(), registerFitnessEffectCallback(), registerMateChoiceCallback(),
registerModifyChildCallback(), registerMutationCallback(), registerMutationEffectCallback(),
registerReproductionCallback(), registerSurvivalCallback(), simulationFinished(),
skipTick(), subsetMutations(), treeSeqCoalesced(), treeSeqOutput(), treeSeqRememberIndividuals(),
treeSeqSimplify()

---

registerReproductionCallback

SLiM method registerReproductionCallback

Description

Documentation for SLiM function `registerReproductionCallback`, which is a method of the SLiM class `Species`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
registerReproductionCallback(id, source, subpop, sex, start, end)
```

Arguments

`id`	An object of type null or integer or string. Must be of length 1 (a singleton). See details for description.
`source`	An object of type string. Must be of length 1 (a singleton). See details for description.
`subpop`	An object of type null or integer or Subpopulation object. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`sex`	An object of type null or string. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`start`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`end`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 727.

Register a block of Eidos source code, represented as the string singleton source, as an Eidos reproduction() callback in the current simulation (specific to the target species), with optional subpopulation subpop (which may be an integer identifier, or NULL, the default, to indicate all subpopulations), optional sex-specificity sex (which may be "M" or "F" in sexual species to make the callback specific to males or females respectively, or NULL for no sex-specificity), and optional start and end ticks all limiting its applicability. The script block will be given identifier id (specified as an integer, or as a string symbolic name such as "s5"); this may be NULL if there is no need to be able to refer to the block later. The registered callback is added to the end of the list of registered SLiMEidosBlock objects, and is active immediately; it may be eligible to execute in the current tick (see section 26.11 for details). The new SLiMEidosBlock will be defined as a global variable immediately by this method (see section 25.12), and will also be returned by this method.

Value

An object of type SLiMEidosBlock object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Species: Sp, addSubpopSplit(), addSubpop(), countOfMutationsOfType(), individualsWithPedigreeID
killIndividuals(), mutationCounts(), mutationFrequencies(), mutationsOfType(),
outputFixedMutations(), outputFull(), outputMutations(), readFromPopulationFile(),
recalculateFitness(), registerFitnessEffectCallback(), registerMateChoiceCallback(),
registerModifyChildCallback(), registerMutationCallback(), registerMutationEffectCallback(),
registerRecombinationCallback(), registerSurvivalCallback(), simulationFinished(),
skipTick(), subsetMutations(), treeSeqCoalesced(), treeSeqOutput(), treeSeqRememberIndividuals(),
treeSeqSimplify()

`registerSurvivalCallback`
SLiM method registerSurvivalCallback

Description

Documentation for SLiM function `registerSurvivalCallback`, which is a method of the SLiM class `Species`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
registerSurvivalCallback(id, source, subpop, start, end)
```

Arguments

`id`	An object of type null or integer or string. Must be of length 1 (a singleton). See details for description.
`source`	An object of type string. Must be of length 1 (a singleton). See details for description.
`subpop`	An object of type null or integer or Subpopulation object. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`start`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`end`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 727.

Register a block of Eidos source code, represented as the string singleton source, as an Eidos survival() callback in the current simulation (specific to the target species), with optional subpopulation subpop (which may be an integer identifier, or NULL, the default, to indicate all subpopulations) and optional start and end ticks all limiting its applicability. The script block will be given identifier id (specified as an integer, or as a string symbolic name such as "s5"); this may be NULL if there is no need to be able to refer to the block later. The registered callback is added to the end of the list of registered SLiMEidosBlock objects, and is active immediately; it may be eligible to execute in the current tick (see section 26.11 for details). The new SLiMEidosBlock will be defined as a global variable immediately by this method (see section 25.12), and will also be returned by this method.

Value

An object of type SLiMEidosBlock object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Species: Sp, addSubpopSplit(), addSubpop(), countOfMutationsOfType(), individualsWithPedigreeID killIndividuals(), mutationCounts(), mutationFrequencies(), mutationsOfType(), outputFixedMutations(), outputFull(), outputMutations(), readFromPopulationFile(), recalculateFitness(), registerFitnessEffectCallback(), registerMateChoiceCallback(), registerModifyChildCallback(), registerMutationCallback(), registerMutationEffectCallback(), registerRecombinationCallback(), registerReproductionCallback(), simulationFinished(), skipTick(), subsetMutations(), treeSeqCoalesced(), treeSeqOutput(), treeSeqRememberIndividuals(), treeSeqSimplify()

---

relatedness *SLiM method relatedness*

Description

Documentation for SLiM function **relatedness**, which is a method of the SLiM class Individual. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
relatedness(individuals)
```

Arguments

individuals	An object of type Individual object. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 683.

Returns a vector containing the degrees of relatedness between the receiver and each of the individuals in individuals. The relatedness between A and B is always 1.0 if A and B are actually the same individual; this facility works even if SLiM's optional pedigree

tracking is not enabled (in which case all other relatedness values will be 0.0). Otherwise, if pedigree tracking is turned on with initializeSLiMOptions(keepPedigrees=T), this method will use the pedigree information described in section 25.7.1 to construct a relatedness estimate. More specifically, this method uses all available pedigree information from the grandparental and parental pedigree records of A and B to compute an estimate of the degree of consanguinity between A and B. Siblings have a relatedness of 0.5, as do parents to their children and vice versa; cousins have a relatedness of 0.125; and so forth. If, according to the pedigree information available, A and B have no blood relationship, the value returned is 0.0. Note that the value returned by relatedness() is what is called the "coefficient of relationship" between the two individuals (Wright, 1922; https:// doi.org/10.1086/279872), and ranges from 0.0 to 1.0. There is another commonly used metric of relatedness, called the "kinship coefficient", that reflects the probability of identity by descent between two individuals A and B. In general, it is approximately equal to one-half of the coefficient of relationship; if an approximate estimate of the kinship coefficient is acceptable, especially in models in which individuals are expected to be outbred, you can simply divide relatedness() by two. However, it should be noted that Wright's coefficient of relationship is not a measure of the probability of identity by descent, and so it is not exactly double the kinship coefficient; they actually measure different things. More precisely, the relationship between them is r = 2 /sqrt((1+fA)(1+fB)), where r is Wright's coefficient of relatedness, is the kinship coefficient, and fA and fB are the inbreeding coefficients of A and B respectively. Note that this relatedness is simply pedigree-based relatedness, and does not necessarily correspond to genetic relatedness, because of the effects of factors like assortment and recombination. If a metric of actual genetic relatedness is desired, tree-sequence recording can be used after simulation is complete, to compute the exact genetic relatedness between individuals based upon the complete ancestry tree (a topic which is beyond the scope of this manual). Actual genetic relatedness cannot presently be calculated during a simulation run; the information is implicitly contained in the recorded tree-sequence tables, but calculating it is too computationally expensive to be reasonable. This method assumes that the grandparents (or the parents, if grandparental information is not available) are themselves unrelated and that they are not inbred; this assumption is necessary because we have no information about their parentage, since SLiM's pedigree tracking information only goes back two generations. Be aware that in a model where inbreeding or selfing occurs at all (including "incidental selfing", where a hermaphroditic individual happens to choose itself as a mate), some level of "background relatedness" will be present and this assumption will be violated. In such circumstances, relatedness() will therefore tend to underestimate the degree of relatedness between individuals, and the greater the degree of inbreeding, the greater the underestimation will be. If inbreeding is allowed in a model - and particularly if it is common - the results of relatedness() should therefore not be taken as an estimate of absolute relatedness, but can still be useful as an estimate of relative relatedness (indicating that, say, A appears from the information available to be more closely related to B than it is to C). See also sharedParentCount() for a different metric of relatedness.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Individual: `In`, `containsMutations()`, `countOfMutationsOfType()`, `setSpatialPosition()`, `sharedParentCount()`, `sumOfMutationsOfType()`, `uniqueMutationsOfType()`

`removeMutations` *SLiM method removeMutations*

Description

Documentation for SLiM function `removeMutations`, which is a method of the SLiM class `Genome`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
removeMutations(mutations, substitute)
```

Arguments

`mutations`	An object of type null or Mutation object. The default value is `NULL`. See details for description.
`substitute`	An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 676.

Remove the mutations in mutations from the target genome(s), if they are present (if they are not present, they will be ignored). If NULL is passed for mutations (which is the default), then all mutations will be removed from the target genomes; in this case, substitute must be F (a specific vector of mutations to be substituted is required). Note that the Mutation objects removed remain valid, and will still be in the simulation's mutation registry (i.e., will be returned by the Species property mutations), until the next tick. Removing

mutations will normally affect the fitness values calculated at the end of the current tick; if you want current fitness values to be affected, you can call the Species method recalculateFitness() - but see the documentation of that method for caveats. The optional parameter substitute was added in SLiM 2.2, with a default of F for backward compatibility. If substitute is T, Substitution objects will be created for all of the removed mutations so that they are recorded in the simulation as having fixed, just as if they had reached fixation and been removed by SLiM's own internal machinery. This will occur regardless of whether the mutations have in fact fixed, regardless of the convertToSubstitution property of the relevant mutation types, and regardless of whether all copies of the mutations have even been removed from the simulation (making it possible to create Substitution objects for mutations that are still segregating). It is up to the caller to perform whatever checks are necessary to preserve the integrity of the simulation's records. Typically substitute will only be set to T in the context of calls like sim.subpopulations.genomes.removeMutations(muts, T), such that the substituted mutations are guaranteed to be entirely removed from circulation. As mentioned above, substitute may not be T if mutations is NULL.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Genome: G, addMutations(), addNewDrawnMutation(), addNewMutation(), containsMarkerMutation(), containsMutations(), countOfMutationsOfType(), mutationCountsInGenomes(), mutationFrequenciesInGe
mutationsOfType(), nucleotides(), outputMS(), outputVCF(), output(), positionsOfMutationsOfType(), readFromMS(), readFromVCF(), sumOfMutationsOfType()

---

removeSpatialMap *SLiM method removeSpatialMap*

Description

Documentation for SLiM function removeSpatialMap, which is a method of the SLiM class Subpopulation. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
removeSpatialMap(map)
```

Arguments

`map`	An object of type string or SpatialMap object. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 743.

Removes the SpatialMap object specified by map from the subpopulation. The parameter map may be either a SpatialMap object, or a string name for spatial map. The map must have been added to the subpopulation with addSpatialMap(); if it has not been, an error results. Removing spatial maps that are no longer in use is optional in most cases. It is generally a good idea because it might decrease SLiM's memory footprint; also, it avoids an error if the subpopulation's spatial bounds are changed (see setSpatialBounds()).

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Subpopulation: `P`, `addCloned()`, `addCrossed()`, `addEmpty()`, `addRecombinant()`, `addSelfed()`, `addSpatialMap()`, `cachedFitness()`, `configureDisplay()`, `defineSpatialMap()`, `outputMSSample()`, `outputSample()`, `outputVCFSample()`, `pointDeviated()`, `pointInBounds()`, `pointPeriodic()`, `pointReflected()`, `pointStopped()`, `pointUniform()`, `removeSubpopulation()`, `sampleIndividuals()`, `setCloningRate()`, `setMigrationRates()`, `setSelfingRate()`, `setSexRatio()`, `setSpatialBounds()`, `setSubpopulationSize()`, `spatialMapColor()`, `spatialMapImage()`, `spatialMapValue()`, `subsetIndividuals()`, `takeMigrants()`

`removeSubpopulation` *SLiM method removeSubpopulation*

Description

Documentation for SLiM function `removeSubpopulation`, which is a method of the SLiM class `Subpopulation`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
removeSubpopulation(void)
```

Arguments

`void`	An object of type . See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 743.

Removes this subpopulation from the model. The subpopulation is immediately removed from the list of active subpopulations, and the symbol representing the subpopulation is undefined. The subpopulation object itself remains unchanged until children are next generated (at which point it is deallocated), but it is no longer part of the simulation and should not be used. Note that this method is only for use in nonWF models, in which there is a distinction between a subpopulation being empty and a subpopulation being removed from the simulation; an empty subpopulation may be re-colonized by migrants, whereas as a removed subpopulation no longer exists at all. WF models do not make this distinction; when a subpopulation is empty it is automatically removed. WF models should therefore call setSubpopulationSize(0) instead of this method; setSubpopulationSize() is the standard way for WF models to change the subpopulation size, including to a size of 0.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (`<bhaller@benhaller.com>`) and Philipp W Messer (`<messer@cornell.edu>`)

See Also

Other Subpopulation: `P`, `addCloned()`, `addCrossed()`, `addEmpty()`, `addRecombinant()`, `addSelfed()`, `addSpatialMap()`, `cachedFitness()`, `configureDisplay()`, `defineSpatialMap()`, `outputMSSample()`, `outputSample()`, `outputVCFSample()`, `pointDeviated()`, `pointInBounds()`, `pointPeriodic()`, `pointReflected()`, `pointStopped()`, `pointUniform()`, `removeSpatialMap()`, `sampleIndividuals()`, `setCloningRate()`, `setMigrationRates()`, `setSelfingRate()`, `setSexRatio()`, `setSpatialBounds()`, `setSubpopulationSize()`, `spatialMapColor()`, `spatialMapImage()`, `spatialMapValue()`, `subsetIndividuals()`, `takeMigrants()`

`reproduction` *SLiM reproduction() callback*

Description

This callback specifies that a code block is providing logic for the production of offspring. It is called once per individual for each generation in which it is active, and is expected to add new individuals to the simulation using the SLiM functions provided for that purpose. This is only used in "nonWF" simulations. For more information on how to use `reproduction()` callback see SLiM Manual: page 606

Usage

```
reproduction(subpop_id, sex)
```

Arguments

`subpop_id`	The id(s) of the subpopulation(s) to which this callback should apply. Can be an integer 1, 2, etc., or character "p1", "p2", etc.
`sex`	The sex of the individuals to apply this callback to.

Details

Global variables available in reproduction callbacks:

individual The focal individual that is expected to reproduce

genome1 One genome of the focal individual

genome2 The other genome of the focal individual

subpop The subpopulation to which the focal individual belongs

Value

None

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other callbacks: early(), first(), fitnessEffect(), fitness(), initialize(), interaction(), late(), mateChoice(), modifyChild(), mutationEffect(), mutation(), recombination(), slim_callbacks(), survival()

Examples

```
slim_block(reproduction(), {

  subpop.addCrossed(individual, subpop.sampleIndividuals(1))

})
```

---

rescale *SLiM method rescale*

Description

Documentation for SLiM function `rescale`, which is a method of the SLiM class SpatialMap. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
rescale(min, max)
```

Arguments

min	An object of type numeric or numeric. Must be of length 1 (a singleton). The default value is 0.0. See details for description.
max	An object of type numeric or numeric. Must be of length 1 (a singleton). The default value is 1.0. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 716.

Rescales the values of the spatial map to the range [min, max]. By default, the rescaling is to the range [0.0, 1.0]. It is required that min be less than max, and that both be finite. Note that the final range may not be exactly [min, max] due to numerical error. The target spatial map is returned, to allow easy chaining of operations.

Value

An object of type SpatialMap object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other SpatialMap: SM, add(), blend(), changeColors(), changeValues(), divide(), exp(), gridValues(), interpolate(), mapColor(), mapImage(), mapValue(), multiply(), power(), range(), sampleImprovedNearbyPoint(), sampleNearbyPoint(), smooth(), subtract()

---

rescheduleScriptBlock

SLiM method rescheduleScriptBlock

Description

Documentation for SLiM function `rescheduleScriptBlock`, which is a method of the SLiM class `Community`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
rescheduleScriptBlock(block, start, end, ticks)
```

Arguments

`block`	An object of type integer or SLiMEidosBlock object. Must be of length 1 (a singleton). See details for description.
`start`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`end`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`ticks`	An object of type null or integer. The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 667.

Reschedule the target script block given by block to execute in a specified set of ticks. The block parameter may be either an integer representing the ID of the desired script block, or a SLiMScriptBlock specified directly. The first way to specify the tick set is with start and end parameter values; block will then execute from start to end, inclusive. In this case, block is returned. The second way to specify the tick set is using the ticks parameter; this is more flexible but more complicated. Since script blocks execute across a contiguous span of ticks defined by their start and end properties, this may result in the duplication of block; one script block will be used for each contiguous span of ticks in ticks. The block object itself will be rescheduled to cover the first such span, whereas duplicates of block will be created to cover subsequent contiguous spans. A vector containing all of the script blocks scheduled by this method, including block, will be returned; this vector is guaranteed to be sorted by the (ascending) scheduled execution order of the blocks. Any duplicates of block created will be given values for the active, source, tag, and type properties equal to the current values for block, but will be given an id of -1 since script block identifiers must be unique; if it is necessary to find the duplicated blocks again later, their tag property should be used. The vector supplied for ticks does not need to be in sorted order, but it must not contain any duplicates. Because this method can create a large number of duplicate script blocks, it can sometimes be better to handle script block scheduling in other ways. If an early() event needs to execute every tenth tick over the whole duration of a long model run, for example, it would not be advisable to use a call like community.rescheduleScriptBlock(s1, ticks=seq(10, 100000, 10)) for that purpose, since that would result in thousands of duplicate script blocks. Instead, it would be preferable to add a test such as if (community.tick event. It is legal to reschedule a script block while the block is executing; a call like community.rescheduleScriptBlock(self, community.tick + 10, community.tick + 10); made inside a given block would therefore also cause the block to execute every tenth tick, although this sort of self-rescheduling code is probably harder to read, maintain, and debug. Whichever way of specifying the tick set is used, the discussion in section 26.11 applies: block may continue to be executed during the current tick cycle stage even after it has been rescheduled, unless it is made inactive using its active property, and similarly, the block may not execute during the current tick cycle stage if it was not already scheduled to do so. Rescheduling script blocks during the tick and tick cycle stage in which they are executing, or in which they are intended to execute, should be avoided. Also, as mentioned in section 23.7, script blocks which are open-ended (i.e., with no specified end tick), are not used in determining whether the end of the simulation has been reached

(because then the simulation would run forever); if you reschedule a block to be open-ended, and to start after the end of the last closed-ended block, the rescheduled block will therefore not run at all (just as such a block would not run at all in other circumstances, too). Note that new script blocks can also be created and scheduled using the register...() methods of Community and Species; by using the same source as a template script block, the template can be duplicated and scheduled for different ticks. In fact, rescheduleScriptBlock() does essentially that internally. In multispecies models, note that all new script blocks created as a side effect of rescheduleScriptBlock() will have the same species and ticks specifier as block; use the register...() methods to create a new block with a different species or ticks specifier.

Value

An object of type SLiMEidosBlock object.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Community: Co, createLogFile(), deregisterScriptBlock(), genomicElementTypesWithIDs(), interactionTypesWithIDs(), mutationTypesWithIDs(), outputUsage(), registerEarlyEvent(), registerFirstEvent(), registerInteractionCallback(), registerLateEvent(), scriptBlocksWithIDs(), simulationFinished(), speciesWithIDs(), subpopulationsWithIDs(), usage()

---

r_inline *Insert R objects into SLiM scripts*

Description

By calling this function inside a slim_block function call you can insert R objects into the script using direct inlining. This function should generally only be used within a slim_block call

Usage

```
r_inline(object, delay = FALSE)

slimr_inline(object, delay = FALSE)
```

Arguments

`object`	R object to inline into the SLiM script.
`delay`	By default `r_inline` will insert the value of `object` into the script when the script is created (e.g. when `slim_script` is called). However, setting `delay = TRUE` will delay the evaluation of `object` until the script is rendered instead (e.g. when `slim_script_render` is called). This allows you to write a `slimr_script` before you have an object available to be inlined (e.g. it allows you to have a 'placeholder' for an object you plan to generate in R later).

Details

Currently supported R objects include all atomic vectors, matrices and arrays and `RasterLayer` objects. Non-atomic vectors like factors are currently not supported and neither are any other special object types, though we plan to support some in the future.

Value

A character vector with the code generated for inlining.

Examples

```
popsizes <- c(100, 200, 300)
slim_script(
 slim_block(initialize(),
        {
          initializeMutationRate(1e-7);
          initializeMutationType("m1", 0.5, "f", 0.0);
          initializeGenomicElementType("g1", m1, 1.0);
          initializeGenomicElement(g1, 0, 99999);
          initializeRecombinationRate(1e-8);
        }),
 slim_block(1,
        {
          popsizes <- r_inline(popsizes)
          sim.addSubpop("p1", popsizes[1]);
          sim.addSubpop("p2", popsizes[2]);
          sim.addSubpop("p3", popsizes[3]);
        }),
 slim_block(10000,
        {
          sim.simulationFinished();
        })
)
```

r_output *Tell SLiM to produce easily parseable output*

Description

Use this function in a slim_block call and it will be converted in the SLiM script into code to make formatted output. This output can easily be read into R and even dynamically read during simulation runs with slim_run from the slimr package. This function should generally only be used within a slim_block call

Usage

```
r_output(
  slimr_expr,
  name,
  do_every = 1,
  send_to = c("data", "file"),
  file_name = tempfile(fileext = ".txt"),
  format = c("csv", "fst"),
  type = NULL,
  expression = NULL,
  time_counter = community.tick
)

slimr_output(
  slimr_expr,
  name,
  do_every = 1,
  send_to = c("data", "file"),
  file_name = tempfile(fileext = ".txt"),
  format = c("csv", "fst"),
  type = NULL,
  expression = NULL,
  time_counter = community.tick
)
```

Arguments

slimr_expr	A SLiM expression to generate output. This can either be a SLiM expression designed to create output, such as outputFull(), or an object created in the SLiM code, in which case r_output will automatically concatenate it to a string and output it
name	The name to use to identify this output.
do_every	How often should the output be produced? Expressed as an integer saying how many generations to run before producing output. e.g. do_every = 10 means to output every 10 generations of the simulation.

`send_to`	***deprecated*** Where to send the output? This could be used to send the output to be stored in the results object ("data"), or to an external file ("file"). This will no longer be available in future versions in favour of an option to have the results stored as a pointer to an external file (to save memory).
`file_name`	***deprecated*** The file name to save output to if `send_to = "file"`
`format`	***deprecated*** The file format to save data if `send_to = "file"`, Only "csv" is implemented but there are plans to support "fst" and "disk.frame".
`type`	Provide a custom type to the output. Used mostly for internal purposes.
`expression`	Provide a custom expression to be included with the output. Used mostly for internal purposes.
`time_counter`	Expression used to extract the simulation time from SLiM. By default this uses the global timing mechanism in SLiM version >4.0: 'community.tick'. For versions <4.0, use 'sim.generation' instead (this parameter is for backwards compatability with old SLiM versions)

Value

An expression with the code to be run in SLiM.

Examples

```
if(slim_is_avail()) {
slim_script(
  slim_block(initialize(),
             {
               initializeMutationRate(1e-7);
               initializeMutationType("m1", 0.5, "f", 0.0);
               initializeGenomicElementType("g1", m1, 1.0);
               initializeGenomicElement(g1, 0, 99999);
               initializeRecombinationRate(1e-8);
             }),
  slim_block(1,
             {
               sim.addSubpop("p1", 100);
             }),
  slim_block(100,
             {
               r_output(p1.outputVCFSample(sampleSize = 10), name = "VCF");
               sim.simulationFinished();
             })
) %>%
slim_run() -> run_w_out

cat(run_w_out$output_data$data[[1]])
}
```

r_output_coords — *Utility function to tell SLiM to output coordinates from spatial simulations*

Description

Utility function to tell SLiM to output coordinates from spatial simulations

Usage

```
r_output_coords(dimensionality = c("x", "xy", "xyz"), ...)

slimr_output_coords(dimensionality = c("x", "xy", "xyz"), ...)
```

Arguments

dimensionality	What dimensionality should be output? Can be "x", "xy", or "xyz".
...	Other arguments to be passed to r_output

Details

Outputs x, y, and z coordinates as separate entries in `slimr_results`, with names "x", "y", and "z".

Value

None

r_output_full — *Utility function to tell SLiM to output its outputFull() output*

Description

Utility function to tell SLiM to output its outputFull() output

Usage

```
r_output_full(name = "full_output", ...)

slimr_output_full(name = "full_output", ...)
```

Arguments

name	Name of output to use to label it in `slimr_results object`. Default is `"full_output"`
...	Other arguments to be passed to r_output

Examples

```
test_sim <- slim_script(
  slim_block_init_minimal(mutation_rate = 1e-6),
  slim_block_add_subpops(1, 100),
  slim_block(1, 20, late(), {
    r_output_full("out", do_every = 10)
  })
)
test_sim
```

---

r_output_nucleotides *Utility function to tell SLiM to output Nucleotides*

Description

Utility function to tell SLiM to output Nucleotides

Usage

```
r_output_nucleotides(
  name = "seqs",
  subpops = FALSE,
  both_genomes = FALSE,
  inds = NULL,
  ...
)

slimr_output_nucleotides(
  name = "seqs",
  subpops = FALSE,
  both_genomes = FALSE,
  inds = NULL,
  ...
)
```

Arguments

name	Name of output to use to label it in slimr_results object. Default is "seqs".
subpops	Should the subpopulation of each sequence be outputted as well?
inds	SLiM expression that returns the individuals to get nucleotides from. By default all individuals are returned.
...	Other arguments to be passed to r_output

Value

None

Examples

```
test_sim <- slim_script(
  slim_block(initialize(), {

    ## tell SLiM to simulate nucleotides
    initializeSLiMOptions(nucleotideBased=T);
    initializeAncestralNucleotides(randomNucleotides(1000));
    initializeMutationTypeNuc("m1", 0.5, "f", 0.0);

    initializeGenomicElementType("g1", m1, 1.0, mmJukesCantor(1e-5));
    initializeGenomicElement(g1, 0, 1000 - 1);
    initializeRecombinationRate(1e-8);

  }),
  slim_block_add_subpops(1, 100),
  slim_block(1, 20, late(), {
    r_output_nucleotides("out", do_every = 10)
  })
)
test_sim
```

---

r_output_sex *Utility function to tell SLim to output sexes of individuals*

Description

Utility function to tell SLim to output sexes of individuals

Usage

```
r_output_sex(name = "sex", ...)

slimr_output_sex(name = "sex", ...)
```

Arguments

`name`	Name of output to use to label it in `slimr_results` `object`. Default is `"sex"`.
`...`	Other arguments to be passed to `r_output`

Value

None

Examples

```
if(slim_is_avail()) {
  test_sim <- slim_script(
    slim_block_init_minimal(mutation_rate = 1e-6),
    slim_block_add_subpops(1, 100),
    slim_block(1, 20, late(), {
      r_output_sex("sex", do_every = 10)
    })
  ) %>%
    slim_run()
  slim_results_to_data(test_sim$output_data)
}
```

---

r_output_snp *Utility function to tell SLiM to output SNP format data*

Description

Use slim_results_to_data on the slim_run results to get this in a nice SNP matrix form.

Usage

```
r_output_snp(name = "snp", subpops = FALSE, ...)

slimr_output_snp(name = "snp", subpops = FALSE, ...)
```

Arguments

name	Name of output to use to label it in slimr_results object. Default is "snp".
subpops	Should the subpopulation of each sequence be outputted as well?
...	Other arguments to be passed to r_output

Value

None

Examples

```
if(slim_is_avail()) {
  test_sim <- slim_script(
    slim_block_init_minimal(mutation_rate = 1e-6),
    slim_block_add_subpops(1, 100),
    slim_block(1, 20, late(), {
      r_output_snp("snp", do_every = 10)
    })
  ) %>%
    slim_run()
  slim_results_to_data(test_sim)
}
```

`r_template` *Create a templated variable*

Description

Create a templated variable inside a SLiM script. This function can be used directly inside a `slim_block` function call, to generate a placemarker that can be replaced with different values dynamically using `slim_script_render`.

Usage

```
r_template(var_name, default = NULL, unquote_strings = FALSE)

slimr_template(var_name, default = NULL, unquote_strings = FALSE)
```

Arguments

`var_name`	Name to use as a placemarker for the variable. This name will be used for replacing values later.
`default`	A default value to be inserted during script rendering if a value is not otherwise provided.
`unquote_strings`	If the value being inserted is of class character, should it be 'unquoted', that is, should double quotes around the value be removed? This is useful when you want to refer to a SLiM object, e.g. to insert `p1.setMigrationRate(...)` instead of `"p1".setMigrationRate(...)`, the latter of which is not valid SLiM code.

Details

Note that this function is only designed to be used inside a `slim_block` function call. If run in any other situation, it won't really do anything, just returning a reference to the placemarker that would have been inserted if run in its correct context.

Value

Returns the placemarker if used outside

`r_template_constant` *Like slimr_template but automatically inserts code to setup variable as a defineConstant() call in the SLiM initialization block.*

Description

Like slimr_template but automatically inserts code to setup variable as a defineConstant() call in the SLiM initialization block.

Usage

```
r_template_constant(var_name, default = NULL, unquote_strings = FALSE)

slimr_template_constant(var_name, default = NULL, unquote_strings = FALSE)
```

Arguments

`var_name`	Name to use as a placemarker for the variable. This name will be used for replacing values later.
`default`	A default value to be inserted during script rendering if a value is not otherwise provided.
`unquote_strings`	If the value being inserted is of class character, should it be 'unquoted', that is, should double quotes around the value be removed? This is useful when you want to refer to a SLiM object, e.g. to insert `p1.setMigrationRate(...)` instead of `"p1".setMigrationRate(...)`, the latter of which is not valid SLiM code.

Details

Note that this function is only designed to be used inside a `slim_block` function call. If run in any other situation, it won't really do anything, just returning a reference to the placemarker that would have been inserted if run in its correct context.

Value

placemarker if used outside 'slim_block'

S *Substitution*

Description

Documentation for Substitution class from SLiM

Details

This class represents a mutation that has been fixed; Mutation objects are converted to Substitution objects upon fixation. Its properties are thus very similar to those of Mutation. Section 1.5.2 presents an overview of the conceptual role of this class. Since Substitution objects represent fixation events that occurred in the past, they are relatively immutable. However, since it may be useful to attach (possibly dynamic) state to substitutions, their tag and subpopID properties are mutable, and they also provide the same getValue() / setValue() functionality as Mutation (inherited from the Eidos class Dictionary). Values set on a Mutation object will carry over to the corresponding Substitution object automatically upon fixation. This class has the following methods (functions):

- None. This class has no methods.

This class has the following properties:

id A property of type integer. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The identifier for this mutation. Each mutation created during a run receives an immutable identifier that will be unique across the duration of the run, and that identifier is carried over to the Substitution object when the mutation fixes.

fixationTick A property of type integer. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The tick in which this mutation fixed.

mutationType A property of type MutationType object. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The MutationType from which this mutation was drawn.

nucleotide A property of type string. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** A string representing the nucleotide associated with this mutation; this will be "A", "C", "G", or "T". If the mutation is not nucleotide-based, this property is unavailable.

nucleotideValue A property of type integer. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** An integer representing the nucleotide associated with this mutation; this will be 0 (A), 1 (C), 2 (G), or 3 (T). If the mutation is not nucleotide-based, this property is unavailable.

originTick A property of type integer. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The tick in which this mutation arose.

position A property of type integer. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The position in the chromosome of this mutation.

selectionCoeff A property of type float. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The selection coefficient of the mutation, drawn from the distribution of fitness effects of its MutationType.

subpopID A property of type integer. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** The identifier of the subpopulation in which this mutation arose. This value is carried over from the Mutation object directly; if a "tag" value was used in the Mutation object (see section 25.10.1), that value will carry over to the corresponding Substitution object. The subpopID in Substitution is a read-write property to allow it to be used as a "tag" in the same way, if the origin subpopulation identifier is not needed.

tag A property of type integer. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** A user-defined integer value. The value of tag is carried over automatically from the original Mutation object. Apart from that, the value of tag is not used by SLiM; it is free for you to use.

`sampleImprovedNearbyPoint`

SLiM method sampleImprovedNearbyPoint

Description

Documentation for SLiM function `sampleImprovedNearbyPoint`, which is a method of the SLiM class `SpatialMap`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
sampleImprovedNearbyPoint(point, maxDistance, functionType, ...)
```

Arguments

`point`	An object of type float. See details for description.
`maxDistance`	An object of type float. Must be of length 1 (a singleton). See details for description.
`functionType`	An object of type string. Must be of length 1 (a singleton). See details for description.
`...`	An object of type NA. NA See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 716.

This variant of sampleNearbyPoint() samples a Metropolis-Hastings move on the spatial map. See sampleNearbyPoint() for discussion of the basic idea. This method proposes a nearby point drawn from the given kernel. If the drawn point has a larger map value than the original point, the new point is returned. If the drawn point has a smaller map value than the original point, it is returned with a probability equal to the ratio between its map value and the original map value, otherwise the original point is returned. The distribution of individuals that move (or not) to new locations governed by this method will converge upon the map itself, in a similar manner to how MCMC converges upon the posterior distribution (assuming no other forces, such as birth or death, influence the distribution of individuals). Movement governed by this method is "improved" in the sense that individuals will tend to remain where they are unless the new sampled point is an improvement for them - a higher map value. Note that unlike sampleNearbyPoint(), this method requires that all map values are non-negative.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other SpatialMap: SM, add(), blend(), changeColors(), changeValues(), divide(), exp(), gridValues(), interpolate(), mapColor(), mapImage(), mapValue(), multiply(), power(), range(), rescale(), sampleNearbyPoint(), smooth(), subtract()

---

sampleIndividuals *SLiM method sampleIndividuals*

Description

Documentation for SLiM function `sampleIndividuals`, which is a method of the SLiM class Subpopulation. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
sampleIndividuals(
  size,
  replace,
  exclude,
  sex,
  minAge,
  maxAge,
  migrant,
  tagL0,
  tagL1,
  tagL2,
  tagL3,
  tagL4
)
```

Arguments

`size`	An object of type integer. Must be of length 1 (a singleton). See details for description.
`replace`	An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.
`exclude`	An object of type null or Individual object. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`sex`	An object of type null or string. Must be of length 1 (a singleton). The default value is `NULL],[Ni$ tag = NULL`. See details for description.
`minAge`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`maxAge`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`migrant`	An object of type null or logical. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`tagL0`	An object of type null or logical. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`tagL1`	An object of type null or logical. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`tagL2`	An object of type null or logical. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`tagL3`	An object of type null or logical. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`tagL4`	An object of type null or logical. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 743.

Returns a vector of individuals, of size less than or equal to parameter size, sampled from the individuals in the target subpopulation. Sampling is done without replacement if replace is F (the default), or with replacement if replace is T. The remaining parameters specify constraints upon the pool of individuals that will be considered candidates for the sampling. Parameter exclude, if non- NULL, may specify a specific individual that should not be considered a candidate (typically the focal individual in some operation). Parameter sex, if non-NULL, may specify a sex ("M" or "F") for the individuals to be drawn, in sexual models. Parameter tag, if non-NULL, may specify a tag property value for the individuals to be drawn. Parameters minAge and maxAge, if non-NULL, may specify a minimum or maximum age for the individuals to be drawn, in nonWF models. Parameter migrant, if non-NULL, may specify a required value for the migrant property of the individuals to be drawn (so T will require that individuals be migrants, F will require that they not be). Finally, parameters tagL0, tagL1, tagL2, tagL3, and tagL4, if non-NULL, may specify a required value (T or F) for the corresponding properties (tagL0, tagL1, tagL2, tagL3, and tagL4) of the individuals to be drawn. Note that if any tag/tagL parameter is specified as non-NULL, that tag/tagL property must have a defined value for every individual in the subpopulation, otherwise an error may result (although this requirement will not necessarily be checked comprehensively by this method in every invocation). If the candidate pool is smaller than the requested sample size, all eligible candidates will be returned (in randomized order); the result will be a zero-length vector if no eligible candidates exist (unlike sample()). This method is similar to getting the individuals property of the subpopulation, using operator [] to select only individuals with the desired properties, and then using sample() to sample from that candidate pool. However, besides being much simpler than the equivalent Eidos code, it is also much faster, and it does not fail if less than the full sample size is available. See subsetIndividuals() for a similar method that returns a full subset, rather than a sample.

Value

An object of type Individual object.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Subpopulation: P, addCloned(), addCrossed(), addEmpty(), addRecombinant(), addSelfed(), addSpatialMap(), cachedFitness(), configureDisplay(), defineSpatialMap(), outputMSSample(), outputSample(), outputVCFSample(), pointDeviated(), pointInBounds(),

pointPeriodic(), pointReflected(), pointStopped(), pointUniform(), removeSpatialMap(), removeSubpopulation(), setCloningRate(), setMigrationRates(), setSelfingRate(), setSexRatio(), setSpatialBounds(), setSubpopulationSize(), spatialMapColor(), spatialMapImage(), spatialMapValue(), subsetIndividuals(), takeMigrants()

`sampleNearbyPoint` *SLiM method sampleNearbyPoint*

Description

Documentation for SLiM function `sampleNearbyPoint`, which is a method of the SLiM class `SpatialMap`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
sampleNearbyPoint(point, maxDistance, functionType, ...)
```

Arguments

`point`	An object of type float. See details for description.
`maxDistance`	An object of type float. Must be of length 1 (a singleton). See details for description.
`functionType`	An object of type string. Must be of length 1 (a singleton). See details for description.
`...`	An object of type NA. NA See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 716.

For a spatial point supplied in point, returns a nearby point sampled from a kernel weighted by the spatial map's values. Only points within the maximum distance of the kernel, maxDistance, will be chosen, and the probability that a given point is chosen will be proportional to the density of the kernel at that point multiplied by the value of the map at that point (interpolated, if interpolation is enabled for the map). Negative values of the map will be treated as zero. The point returned will be within spatial bounds, respecting periodic boundaries if in effect (so there is no need to call pointPeriodic() on the result). The kernel is specified with a kernel shape, functionType, followed by zero or more ellipsis arguments; see smooth() for further information. For this method, at present only kernel types "f", "l", "e", "n", and "t" are supported, and type "t" is not presently supported for 3D kernels. This method can be used to find points in the vicinity of individuals that are favorable - possessing more resources, or better environmental conditions, etc. It can also be used to guide the dispersal or foraging behavior of individuals. See sampleImprovedNearbyPoint() for a variant that may be useful for directed movement across a landscape.

Note that the algorithm for sampleNearbyPoint() works by rejection sampling, and so will be very inefficient if the maximum value of the map (anywhere, across the entire map) is much larger than the typical value of the map where individuals are. The algorithm for sampleImprovedNearbyPoint() is different, and does not exhibit this performance issue.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other SpatialMap: SM, add(), blend(), changeColors(), changeValues(), divide(), exp(), gridValues(), interpolate(), mapColor(), mapImage(), mapValue(), multiply(), power(), range(), rescale(), sampleImprovedNearbyPoint(), smooth(), subtract()

SB *SLiMBuiltin*

Description

Documentation for SLiMBuiltin class from SLiM

Details

SLiM provides a small number of built-in functions, documented here. Note that these are distinct from the functions built into the Eidos language itself, which are documented in the Eidos manual. This class has the following methods (functions):

- codonsToAminoAcids
- mm16To256
- mmJukesCantor
- mmKimura
- nucleotideCounts
- nucleotideFrequencies
- nucleotidesToCodons

- calcFST
- calcHeterozygosity
- calcInbreedingLoad
- calcPairHeterozygosity
- calcVA
- calcWattersonsTheta
- summarizeIndividuals
- treeSeqMetadata

This class has the following properties:

None. This class has no properties.

See Also

Other SLiMBuiltin: calcFST(), calcHeterozygosity(), calcInbreedingLoad(), calcPairHeterozygosity(), calcVA(), calcWattersonsTheta(), codonsToAminoAcids(), mm16To256(), mmJukesCantor(), mmKimura(), nucleotideCounts(), nucleotideFrequencies(), nucleotidesToCodons(), summarizeIndividuals(), treeSeqMetadata()

---

scriptBlocksWithIDs *SLiM method scriptBlocksWithIDs*

Description

Documentation for SLiM function `scriptBlocksWithIDs`, which is a method of the SLiM class `Community`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
scriptBlocksWithIDs(ids)
```

Arguments

`ids`	An object of type integer. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 668.

Find and return the SLiMEidosBlock objects with id values matching the values in ids. If no matching SLiMEidosBlock object can be found with a given id, an error results.

Value

An object of type SLiMEidosBlock object.

Copyright

This is documentation for a function in the SLiM software, and has been reproduced from the official manual, which can be found here: http://benhaller.com/slim/SLiM_Manual.pdf. This documentation is Copyright © 2016-2020 Philipp Messer. All rights reserved. More information about SLiM can be found on the official website: https://messerlab.org/slim/

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Community: Co, createLogFile(), deregisterScriptBlock(), genomicElementTypesWithIDs(), interactionTypesWithIDs(), mutationTypesWithIDs(), outputUsage(), registerEarlyEvent(), registerFirstEvent(), registerInteractionCallback(), registerLateEvent(), rescheduleScriptBlock(), simulationFinished(), speciesWithIDs(), subpopulationsWithIDs(), usage()

SEB *SLiMEidosBlock*

Description

Documentation for SLiMEidosBlock class from SLiM

Details

This class represents a block of Eidos code registered in a SLiM simulation. All Eidos events and Eidos callbacks defined in the SLiM input file of the current simulation are instantiated as SLiMEidosBlock objects and are available through the allScriptBlocks property of Community and the scriptBlocks property of Species; see sections 25.3.1 and 25.15.1. In addition, new script blocks can be created programmatically and registered with the simulation, and registered script blocks can be deregistered; see the -register...() and -deregisterScriptBlock() methods of Community and Species in sections 25.3.2 and 25.15.2. The currently executing script block is available through the self global; see section 26.11. This class has the following methods (functions):

- None. This class has no methods.

This class has the following properties:

active A property of type integer. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** If this evaluates to logical F (i.e., is equal to 0), the script block is inactive and will not be called. The value of active for all registered script blocks is reset to -1 at the beginning of each tick, prior to script events being called, thus activating all blocks (except callbacks associated with a species that is not active in that tick, which are deactivated as part of the deactivation of the species). Any integer value other than -1 may be used instead of -1 to represent that a block is active; for example, active may be used as a counter to make a block execute a fixed number of times in each tick. This value is not cached by SLiM; if it is changed, the new value takes effect immediately. For example, a callback might be activated and inactivated repeatedly during a single tick.

end A property of type integer. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The last tick in which the script block is active.

id A property of type integer. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The identifier for this script block; for script s3, for example, this is 3. A script block for which no id was given will have an id of -1.

source A property of type string. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The source code string of the script block.

speciesSpec A property of type Species object. This property is a constant, so it is not modifiable. **Property Description:** The species specifier for the script block. The species specifier for a callback block indicates the callback's associated species; the callback is called to modify the default behavior for that species. If the script block has no species specifier, this property's value is a zero-length object vector of class Species. This property is read-only; normally it is set by preceding the definition of a callback with a species specifier, of the form species <species-name>.

start A property of type integer. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The first tick in which the script block is active.

tag A property of type integer. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** A user-defined integer value. The value of tag is initially undefined, and it is an error to try to read it; if you wish it to have a defined value, you must arrange that yourself by explicitly setting its value prior to using it elsewhere in your code. The value of tag is not used by SLiM; it is free for you to use.

ticksSpec A property of type Species object. This property is a constant, so it is not modifiable. **Property Description:** The ticks specifier for the script block. The ticks specifier for an event block indicates the event's associated species; the event executes only in ticks when that species is active. If the script block has no ticks specifier, this property's value is a zero-length object vector of class Species. This property is read-only; normally it is set by preceding the definition of an event with a ticks specifier, of the form ticks <species-name>.

type A property of type string. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The type of the script block; this

will be ”first”, ”early”, or ”late” for the three types of Eidos ”mutation”, ”mutationEffect”, ”recombination”, ”reproduction”, or ”survival” for the respective types of Eidos callbacks (see section 25.1 and chapter 26).

`setAncestralNucleotides`

SLiM method setAncestralNucleotides

Description

Documentation for SLiM function `setAncestralNucleotides`, which is a method of the SLiM class `Chromosome`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
setAncestralNucleotides(sequence)
```

Arguments

`sequence`	An object of type integer or string. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 662.

This method, which may be called only in nucleotide-based models (see section 1.8), replaces the ancestral nucleotide sequence for the model. The sequence parameter is interpreted exactly as it is in the initializeAncestralSequence() function; see that documentation for details (section 25.1). The length of the ancestral sequence is returned. It is unusual to replace the ancestral sequence in a running simulation, since the nucleotide states of segregating and fixed mutations will depend upon the original ancestral sequence. It can be useful when loading a new population state with readFromMS() or readFromVCF(), such as when resetting the simulation state to an earlier state in a conditional simulation; however, that is more commonly done using readFromPopulationFile() with a SLiM or .trees file.

Value

An object of type integer. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Chromosome: Ch, ancestralNucleotides(), drawBreakpoints(), setGeneConversion(), setHotspotMap(), setMutationRate(), setRecombinationRate()

---

setCloningRate *SLiM method setCloningRate*

Description

Documentation for SLiM function setCloningRate, which is a method of the SLiM class Subpopulation. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
setCloningRate(rate)
```

Arguments

rate	An object of type numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 744.

Set the cloning rate of this subpopulation. The rate is changed to rate, which should be between 0.0 and 1.0, inclusive. Clonal reproduction can be enabled in both non-sexual (i.e. hermaphroditic) and sexual simulations. In non-sexual simulations, rate must be a singleton value representing the overall clonal reproduction rate for the subpopulation. In sexual simulations, rate may be either a singleton (specifying the clonal reproduction rate for both sexes) or a vector containing two numeric values (the female and male cloning rates specified separately, at indices 0 and 1 respectively). During mating and offspring generation, the probability that any given offspring individual will be generated by cloning - by asexual reproduction without gametes or meiosis - will be equal to the cloning rate (for its sex, in sexual simulations) set in the parental (not the offspring!) subpopulation.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Subpopulation: P, addCloned(), addCrossed(), addEmpty(), addRecombinant(), addSelfed(), addSpatialMap(), cachedFitness(), configureDisplay(), defineSpatialMap(), outputMSSample(), outputSample(), outputVCFSample(), pointDeviated(), pointInBounds(), pointPeriodic(), pointReflected(), pointStopped(), pointUniform(), removeSpatialMap(), removeSubpopulation(), sampleIndividuals(), setMigrationRates(), setSelfingRate(), setSexRatio(), setSpatialBounds(), setSubpopulationSize(), spatialMapColor(), spatialMapImage(), spatialMapValue(), subsetIndividuals(), takeMigrants()

---

setConstraints *SLiM method setConstraints*

Description

Documentation for SLiM function `setConstraints`, which is a method of the SLiM class `InteractionType`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
setConstraints(
  who,
  sex,
  tag,
  minAge,
  maxAge,
  migrant,
  tagL0,
  tagL1,
  tagL2,
  tagL3,
  tagL4
)
```

Arguments

`who`	An object of type string. Must be of length 1 (a singleton). See details for description.
`sex`	An object of type null or string. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`tag`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`minAge`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`maxAge`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`migrant`	An object of type null or logical. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`tagL0`	An object of type null or logical. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`tagL1`	An object of type null or logical. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`tagL2`	An object of type null or logical. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`tagL3`	An object of type null or logical. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`tagL4`	An object of type null or logical. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 697.

Sets constraints upon which individuals can be receivers and/or exerters, making the target InteractionType measure interactions between only subsets of the population. The parameter who specifies upon whom the specified constraints apply; it may be ”exerters” to set constraints upon exerters, ”receivers” to set constraints upon receivers, or ”both” to set constraints upon both. If ”both” is used, the same constraints are set for both exerters and receivers; different constraints can be set for exerters versus receivers by making a separate call to setConstraints() for each. Constraints only affect queries that involve the concept of interaction; for example, they will affect the result of nearestInteractingNeighbors(), but not the result of nearestNeighbors(). The constraints specified by a given call to setConstraints() override all previously set constraints for the category specified (receivers, exerter, or both). There is a general policy for the remaining arguments: they are NULL by default, and if NULL is used, it specifies ”no constraint” for that property (removing any currently existing constraint for that property). The sex parameter constrains the sex of individuals; it may be ”M” or ”F” (or ”*” as another way of specifying no constraint, for historical reasons). If sex is ”M” or ”F”, the individuals to which the constraint is applied (potential receivers/exerters) must belong to a sexual species. The tag parameter constrains the tag property of individuals; if this set, the individuals to which the constraint is applied must have defined tag values. The minAge and maxAge properties constrain the age property

of individuals to the given minimum and/or maximum values; these constraints can only be used in nonWF models. The migrant property constraints the migrant property of individuals (T constrains to only migrants, F to only non-migrants). Finally, the tagL0, tagL1, tagL2, tagL3, and tagL4 properties constrain the corresponding logical properties of individuals, requiring them to be either T or F as specified; the individuals to which these constraints are applied must have defined values for the constrained property or properties. Again, NULL should be supplied (as it is by default) for any property which you do not wish to constrain. These constraints may be used in any combination, as desired. For example, calling setConstraints("receivers", sex="M", minAge=5, tagL0=T) constrains the interaction type's operation so that receivers must be males, with an age of at least 5, with a tagL0 property value of T. For that configuration the potential receivers used with the interaction type must be sexual (since sex is specified), must be in a nonWF model (since minAge is specified), and must have a defined value for their tagL0 property (since that property is constrained). Note that the sexSegregation parameter to initializeInteractionType() is a shortcut which does the same thing as the corresponding calls to setConstraints(). Exerter constraints are applied at evaluate() time, whereas receiver constraints are applied at query time; see the InteractionType class documentation (section 25.8) for further discussion. The interaction constraints for an interaction type are normally a constant in simulations; in any case, they cannot be changed when an interaction has already been evaluated, so either they should be set prior to evaluation, or unevaluate() should be called first.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other InteractionType: IT, clippedIntegral(), distanceFromPoint(), distance(), drawByStrength(), evaluate(), interactingNeighborCount(), interactionDistance(), localPopulationDensity(), nearestInteractingNeighbors(), nearestNeighborsOfPoint(), nearestNeighbors(), neighborCountOfPoint(), neighborCount(), setInteractionFunction(), strength(), testConstraints(), totalOfNeighborStrengths(), unevaluate()

`setDistribution` *SLiM method setDistribution*

Description

Documentation for SLiM function `setDistribution`, which is a method of the SLiM class `MutationType`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
setDistribution(distributionType, ...)
```

Arguments

`distributionType`	An object of type string. Must be of length 1 (a singleton). See details for description.
...	An object of type NA. NA See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 709.

Set the distribution of fitness effects for a mutation type. The distributionType may be ”f”, in which case the ellipsis ... should supply a numeric$ fixed selection coefficient; ”e”, in which case the ellipsis should supply a numeric$ mean selection coefficient for the exponential distribution; ”g”, in which case the ellipsis should supply a numeric$ mean selection coefficient and a numeric$ alpha shape parameter for a gamma distribution; ”n”, in which case the ellipsis should supply a numeric$ mean selection coefficient and a numeric$ sigma (standard deviation) parameter for a normal distribution; ”p”, in which case the ellipsis should supply a numeric$ mean selection coefficient and a numeric$ scale parameter for a Laplace distribution; ”w”, in which case the ellipsis should supply a numeric$ scale parameter and a numeric$ k shape parameter for a Weibull distribution; or ”s”, in which case the ellipsis should supply a string$ Eidos script parameter. See section 25.11 above for discussions of these distributions and their uses. The DFE for a mutation type is normally a constant in simulations, so be sure you know what you are doing.

Value

An object of type float or void.

Copyright

More information about SLiM can be found on the official website: https://messerlab.org/slim/

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other MutationType: MT, drawSelectionCoefficient()

`setFilePath` *SLiM method setFilePath*

Description

Documentation for SLiM function `setFilePath`, which is a method of the SLiM class LogFile. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
setFilePath(filePath, initialContents, append, compress, sep)
```

Arguments

`filePath`	An object of type string. Must be of length 1 (a singleton). See details for description.
`initialContents`	An object of type null or string. The default value is `NULL`. See details for description.
`append`	An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.
`compress`	An object of type null or logical. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`sep`	An object of type null or string. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 703.

Redirects the LogFile to write new rows to a new filePath. Any rows that have been buffered but not flushed will be written to the previous file first, as if flush() had been called. With this call, new initialContents may be supplied, which will either replace any existing file or will be appended to it, depending upon the value of append. New values may be supplied

for compress and sep; the meaning of these parameters is identical to their meaning in createLogFile(), except that a value of NULL for these means "do not change this setting from its previous value". In effect, then, this method lets you start a completely new log file at a new path, without having to create and configure a new LogFile object. The new file will be created (or appended) synchronously, with the specified initial contents.

Value

An object of type void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or logical.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other LogFile: LF, addCustomColumn(), addCycleStage(), addCycle(), addKeysAndValuesFrom(), addMeanSDColumns(), addPopulationSexRatio(), addPopulationSize(), addSubpopulationSexRatio(), addSubpopulationSize(), addSuppliedColumn(), addTick(), clearKeysAndValues(), flush(), logRow(), setLogInterval(), setSuppliedValue(), setValue(), willAutolog()

---

setGeneConversion *SLiM method setGeneConversion*

Description

Documentation for SLiM function setGeneConversion, which is a method of the SLiM class Chromosome. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
setGeneConversion(
  nonCrossoverFraction,
  meanLength,
  simpleConversionFraction,
  bias
)
```

Arguments

`nonCrossoverFraction`
: An object of type numeric or numeric or numeric or numeric. Must be of length 1 (a singleton). See details for description.

`meanLength`
: An object of type numeric or numeric or numeric or numeric. Must be of length 1 (a singleton). See details for description.

`simpleConversionFraction`
: An object of type numeric or numeric or numeric or numeric. Must be of length 1 (a singleton). See details for description.

`bias`
: An object of type numeric or numeric or numeric or numeric. Must be of length 1 (a singleton). The default value is `0`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 662.

This method switches the recombination model to the "double-stranded break (DSB)" model (if it is not already set to that), and configures the details of the gene conversion tracts that will therefore be modeled (see section 1.5.6 for discussion of the "DSB" recombination model). The meanings and effects of the parameters exactly mirror the initializeGeneConversion() function; see section 25.1 for details.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (`<bhaller@benhaller.com>`) and Philipp W Messer (`<messer@cornell.edu>`)

See Also

Other Chromosome: `Ch`, `ancestralNucleotides()`, `drawBreakpoints()`, `setAncestralNucleotides()`, `setHotspotMap()`, `setMutationRate()`, `setRecombinationRate()`

`setGenomicElementType`

SLiM method setGenomicElementType

Description

Documentation for SLiM function `setGenomicElementType`, which is a method of the SLiM class `GenomicElement`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
setGenomicElementType(genomicElementType)
```

Arguments

`genomicElementType`
: An object of type integer or GenomicElementType object. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 677.

Set the genomic element type used for a genomic element (see section 1.5.4). The genomicElementType parameter should supply the new genomic element type for the element, either as a GenomicElementType object or as an integer identifier. The genomic element type for a genomic element is normally a constant in simulations, so be sure you know what you are doing.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other GenomicElement: `GE`

`setHotspotMap` *SLiM method setHotspotMap*

Description

Documentation for SLiM function `setHotspotMap`, which is a method of the SLiM class `Chromosome`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
setHotspotMap(multipliers, ends, sex)
```

Arguments

`multipliers`	An object of type numeric. See details for description.
`ends`	An object of type null or integer. The default value is `NULL`. See details for description.
`sex`	An object of type string. Must be of length 1 (a singleton). The default value is `"*"`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 662.

In nucleotide-based models, set the mutation rate multiplier along the chromosome. There are two ways to call this method. If the optional ends parameter is NULL (the default), then multipliers must be a singleton value that specifies a single multiplier to be used along the entire chromosome. If, on the other hand, ends is supplied, then multipliers and ends must be the same length, and the values in ends must be specified in ascending order. In that case, multipliers and ends taken together specify the multipliers to be used along successive contiguous stretches of the chromosome, from beginning to end; the last position specified in ends should extend to the end of the chromosome (as previously determined, during simulation initialization). See the initializeHotspotMap() function for further discussion of precisely how these multipliers and positions are interpreted. If the optional sex parameter is "*" (the default), then the supplied hotspot map will be used for both sexes (which is the only option for hermaphroditic simulations). In sexual simulations sex may be "M" or "F" instead, in which case the supplied hotspot map is used only for that sex. Note that whether sex-specific hotspot maps will be used is set by the way that the simulation is initially configured with initializeHotspot(), and cannot be changed with this method; so if the simulation was set up to use sex-specific hotspot maps then sex must be "M" or "F" here, whereas if it was set up not to, then sex must be "*" or unsupplied here. If

a simulation needs sex-specific hotspot maps only some of the time, the male and female maps can simply be set to be identical the rest of the time. The hotspot map is normally constant in simulations, so be sure you know what you are doing.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Chromosome: Ch, ancestralNucleotides(), drawBreakpoints(), setAncestralNucleotides(), setGeneConversion(), setMutationRate(), setRecombinationRate()

---

setInteractionFunction

SLiM method setInteractionFunction

Description

Documentation for SLiM function `setInteractionFunction`, which is a method of the SLiM class InteractionType. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
setInteractionFunction(functionType, ...)
```

Arguments

functionType	An object of type string. Must be of length 1 (a singleton). See details for description.
...	An object of type NA. NA See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 698.

Set the function used to translate spatial distances into interaction strengths for an interaction type. The functionType may be "f", in which case the ellipsis ... should supply a numeric$ fixed interaction strength; "l", in which case the ellipsis should supply a numeric$ maximum strength for a linear function; "e", in which case the ellipsis should supply a numeric$ maximum strength and a numeric$ lambda (rate) parameter for a negative exponential function; "n", in which case the ellipsis should supply a numeric$ maximum strength and a numeric$ sigma (standard deviation) parameter for a Gaussian function; "c", in which case the ellipsis should supply a numeric$ maximum strength and a numeric$ scale parameter for a Cauchy distribution function; or "t", in which case the ellipsis should supply a numeric$ maximum strength, a numeric$ degrees of freedom, and a numeric$ scale parameter for a t-distribution function. See section 25.8 above for discussions of these interaction functions. Non-spatial interactions must use function type "f", since no distance values are available in that case. The interaction function for an interaction type is normally a constant in simulations; in any case, it cannot be changed when an interaction has already been evaluated, so either it should be set prior to evaluation, or unevaluate() should be called first.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other InteractionType: IT, clippedIntegral(), distanceFromPoint(), distance(), drawByStrength(), evaluate(), interactingNeighborCount(), interactionDistance(), localPopulationDensity(), nearestInteractingNeighbors(), nearestNeighborsOfPoint(), nearestNeighbors(), neighborCountOfPoint(), neighborCount(), setConstraints(), strength(), testConstraints(), totalOfNeighborStrengths(), unevaluate()

`setLogInterval` *SLiM method setLogInterval*

Description

Documentation for SLiM function `setLogInterval`, which is a method of the SLiM class `LogFile`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
setLogInterval(logInterval)
```

Arguments

`logInterval`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 703.

Sets the automatic logging interval. A logInterval of NULL stops automatic logging immediately. Other values request that a new row should be logged (as if logRow() were called) at the end of every logInterval ticks (just before the tick count increment, in both WF and nonWF models), starting at the end of the tick in which setLogInterval() was called.

Value

An object of type void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or logical.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other LogFile: `LF`, `addCustomColumn()`, `addCycleStage()`, `addCycle()`, `addKeysAndValuesFrom()`, `addMeanSDColumns()`, `addPopulationSexRatio()`, `addPopulationSize()`, `addSubpopulationSexRatio()`, `addSubpopulationSize()`, `addSuppliedColumn()`, `addTick()`, `clearKeysAndValues()`, `flush()`, `logRow()`, `setFilePath()`, `setSuppliedValue()`, `setValue()`, `willAutolog()`

`setMigrationRates` *SLiM method setMigrationRates*

Description

Documentation for SLiM function `setMigrationRates`, which is a method of the SLiM class `Subpopulation`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
setMigrationRates(sourceSubpops, rates)
```

Arguments

`sourceSubpops`	An object of type integer or Subpopulation object. See details for description.
`rates`	An object of type numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 744.

Set the migration rates to this subpopulation from the subpopulations in sourceSubpops to the corresponding rates specified in rates; in other words, rates gives the expected fractions of the children in this subpopulation that will subsequently be generated from parents in the subpopulations sourceSubpops (see section 23.2.1). This method will only set the migration fractions from the subpopulations given; migration rates from other subpopulations will be left unchanged (explicitly set a zero rate to turn off migration from a given subpopulation). The type of sourceSubpops may be either integer, specifying subpopulations by identifier, or object, specifying subpopulations directly.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Subpopulation: P, addCloned(), addCrossed(), addEmpty(), addRecombinant(), addSelfed(), addSpatialMap(), cachedFitness(), configureDisplay(), defineSpatialMap(), outputMSSample(), outputSample(), outputVCFSample(), pointDeviated(), pointInBounds(), pointPeriodic(), pointReflected(), pointStopped(), pointUniform(), removeSpatialMap(), removeSubpopulation(), sampleIndividuals(), setCloningRate(), setSelfingRate(), setSexRatio(), setSpatialBounds(), setSubpopulationSize(), spatialMapColor(), spatialMapImage(), spatialMapValue(), subsetIndividuals(), takeMigrants()

---

setMutationFractions *SLiM method setMutationFractions*

Description

Documentation for SLiM function setMutationFractions, which is a method of the SLiM class GenomicElementType. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
setMutationFractions(mutationTypes, proportions)
```

Arguments

mutationTypes	An object of type integer or MutationType object. See details for description.
proportions	An object of type numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 678.

Set the mutation type fractions contributing to a genomic element type. The mutationTypes vector should supply the mutation types used by the genomic element (either as MutationType objects or as integer identifiers), and the proportions vector should be of equal length, specifying the relative proportion of mutations that will be drawn from each corresponding type (see section 1.5.4). This is normally a constant in simulations, so be sure you know what you are doing.

Value

An object of type void or void.

Copyright

Author(s)

Benjamin C Haller (`<bhaller@benhaller.com>`) and Philipp W Messer (`<messer@cornell.edu>`)

See Also

Other GenomicElementType: `GET`, `setMutationMatrix()`

`setMutationMatrix` *SLiM method setMutationMatrix*

Description

Documentation for SLiM function `setMutationMatrix`, which is a method of the SLiM class `GenomicElementType`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
setMutationMatrix(mutationMatrix)
```

Arguments

`mutationMatrix`
: An object of type float. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 678.

Sets a new nucleotide mutation matrix for the genomic element type. This replaces the mutation matrix originally set by initializeGenomicElementType(). This method may only be called in nucleotide-based models.

Value

An object of type void or void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other GenomicElementType: GET, setMutationFractions()

---

setMutationRate *SLiM method setMutationRate*

Description

Documentation for SLiM function setMutationRate, which is a method of the SLiM class Chromosome. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
setMutationRate(rates, ends, sex)
```

Arguments

rates	An object of type numeric. See details for description.
ends	An object of type null or integer. The default value is NULL. See details for description.
sex	An object of type string. Must be of length 1 (a singleton). The default value is "*". See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 663.

Set the mutation rate per base position per gamete. There are two ways to call this method. If the optional ends parameter is NULL (the default), then rates must be a singleton value that specifies a single mutation rate to be used along the entire chromosome. If, on the other hand, ends is supplied, then rates and ends must be the same length, and the values in ends must be specified in ascending order. In that case, rates and ends taken together specify the mutation rates to be used along successive contiguous stretches of the chromosome, from beginning to end; the last position specified in ends should extend to the end of the chromosome (as previously determined, during simulation initialization). See the initializeMutationRate() function for further discussion of precisely how these rates and positions are interpreted. If the optional sex parameter is "*" (the default), then the supplied mutation rate map will be used for both sexes (which is the only option for hermaphroditic simulations). In sexual simulations sex may be "M" or "F" instead, in which case the supplied mutation rate map is used only for that sex. Note that whether sex-specific mutation rate maps will be used is set by the way that the simulation is initially configured with initializeMutationRate(), and cannot be changed with this method; so if the simulation was set up to use sex-specific mutation rate maps then sex must be "M" or "F" here, whereas if it was set up not to, then sex must be "*" or unsupplied here. If a simulation needs sexspecific mutation rate maps only some of the time, the male and female maps can simply be set to be identical the rest of the time. The mutation rate intervals are normally a constant in simulations, so be sure you know what you are doing. In nucleotide-based models, setMutationRate() may not be called. If variation in the mutation rate along the chromosome is desired, setHotspotMap() should be used.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Chromosome: Ch, ancestralNucleotides(), drawBreakpoints(), setAncestralNucleotides(), setGeneConversion(), setHotspotMap(), setRecombinationRate()

`setMutationType` *SLiM method setMutationType*

Description

Documentation for SLiM function `setMutationType`, which is a method of the SLiM class `Mutation`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
setMutationType(mutType)
```

Arguments

`mutType`	An object of type integer or MutationType object. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 705.

Set the mutation type of the mutation to mutType (which may be specified as either an integer identifier or a MutationType object). This implicitly changes the dominance coefficient of the mutation to that of the new mutation type, since the dominance coefficient is a property of the mutation type. On the other hand, the selection coefficient of the mutation is not changed, since it is a property of the mutation object itself; it can be changed explicitly using the setSelectionCoeff() method if so desired. The mutation type of a mutation is normally a constant in simulations, so be sure you know what you are doing. Changing this will normally affect the fitness values calculated toward the end of the current tick; if you want current fitness values to be affected, you can call the Species method recalculateFitness() - but see the documentation of that method for caveats. In nucleotide-based models, a restriction applies: nucleotide-based mutations may not be changed to a non-nucleotide-based mutation type, and non-nucleotide-based mutations may not be changed to a nucleotide-based mutation type.

Value

An object of type void or void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Mutation: `M`, `setSelectionCoeff()`

`setRecombinationRate` *SLiM method setRecombinationRate*

Description

Documentation for SLiM function `setRecombinationRate`, which is a method of the SLiM class `Chromosome`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
setRecombinationRate(rates, ends, sex)
```

Arguments

`rates`	An object of type numeric. See details for description.
`ends`	An object of type null or integer. The default value is `NULL`. See details for description.
`sex`	An object of type string. Must be of length 1 (a singleton). The default value is `"*"`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 663.

Set the recombination rate per base position per gamete. All rates must be in the interval [0.0, 0.5]. There are two ways to call this method. If the optional ends parameter is NULL (the default), then rates must be a singleton value that specifies a single recombination rate to be used along the entire chromosome. If, on the other hand, ends is supplied, then rates and ends must be the same length, and the values in ends must be specified in ascending order. In that case, rates and ends taken together specify the recombination rates to be used along successive contiguous stretches of the chromosome, from beginning to end; the last position specified in ends should extend to the end of the chromosome (as previously determined, during simulation initialization). See the initializeRecombinationRate() function for further discussion of precisely how these rates and positions are interpreted. If the optional sex parameter is "*" (the default), then the supplied recombination rate map will be used for both sexes (which is the only option for hermaphroditic simulations). In sexual simulations sex may be "M" or "F" instead, in which case the supplied recombination map is used only for that sex. Note that whether sex-specific recombination maps will be used

is set by the way that the simulation is initially configured with initializeRecombinationRate(), and cannot be changed with this method; so if the simulation was set up to use sex-specific recombination maps then sex must be "M" or "F" here, whereas if it was set up not to, then sex must be "*" or unsupplied here. If a simulation needs sex-specific recombination maps only some of the time, the male and female maps can simply be set to be identical the rest of the time. The recombination intervals are normally a constant in simulations, so be sure you know what you are doing.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Chromosome: Ch, ancestralNucleotides(), drawBreakpoints(), setAncestralNucleotides(), setGeneConversion(), setHotspotMap(), setMutationRate()

---

setSelectionCoeff *SLiM method setSelectionCoeff*

Description

Documentation for SLiM function `setSelectionCoeff`, which is a method of the SLiM class Mutation. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
setSelectionCoeff(selectionCoeff)
```

Arguments

`selectionCoeff`
: An object of type float. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 705.

Set the selection coefficient of the mutation to selectionCoeff. The selection coefficient will be changed for all individuals that possess the mutation, since they all share a single Mutation object (note that the dominance coefficient will remain unchanged, as it is determined by the mutation type). This is normally a constant in simulations, so be sure you know what you are doing; often setting up a mutationEffect() callback (see section 26.2) is preferable, in order to modify the selection coefficient in a more limited and controlled fashion (see section 10.5 for further discussion of this point). Changing this will normally affect the fitness values calculated toward the end of the current tick; if you want current fitness values to be affected, you can call the Species method recalculateFitness() - but see the documentation of that method for caveats.

Value

An object of type void or void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Mutation: M, setMutationType()

---

setSelfingRate *SLiM method setSelfingRate*

Description

Documentation for SLiM function `setSelfingRate`, which is a method of the SLiM class Subpopulation. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
setSelfingRate(rate)
```

Arguments

`rate`	An object of type numeric. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 744.

Set the selfing rate of this subpopulation. The rate is changed to rate, which should be between 0.0 and 1.0, inclusive. Selfing can only be enabled in non-sexual (i.e. hermaphroditic) simulations. During mating and offspring generation, the probability that any given offspring individual will be generated by selfing - by self-fertilization via gametes produced by meiosis by a single parent - will be equal to the selfing rate set in the parental (not the offspring!) subpopulation (see section 23.2.1).

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Subpopulation: P, addCloned(), addCrossed(), addEmpty(), addRecombinant(), addSelfed(), addSpatialMap(), cachedFitness(), configureDisplay(), defineSpatialMap(), outputMSSample(), outputSample(), outputVCFSample(), pointDeviated(), pointInBounds(), pointPeriodic(), pointReflected(), pointStopped(), pointUniform(), removeSpatialMap(), removeSubpopulation(), sampleIndividuals(), setCloningRate(), setMigrationRates(), setSexRatio(), setSpatialBounds(), setSubpopulationSize(), spatialMapColor(), spatialMapImage(), spatialMapValue(), subsetIndividuals(), takeMigrants()

---

setSexRatio *SLiM method setSexRatio*

Description

Documentation for SLiM function `setSexRatio`, which is a method of the SLiM class `Subpopulation`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
setSexRatio(sexRatio)
```

Arguments

`sexRatio`	An object of type float. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 744.

Set the sex ratio of this subpopulation to sexRatio. As defined in SLiM, this is actually the fraction of the subpopulation that is male; in other words, the M:(M+F) ratio. This will take effect when children are next generated; it does not change the current subpopulation state. Unlike the selfing rate, the cloning rate, and migration rates, the sex ratio is deterministic: SLiM will generate offspring that exactly satisfy the requested sex ratio (within integer roundoff limits). See section 23.2.1 for further details.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (`<bhaller@benhaller.com>`) and Philipp W Messer (`<messer@cornell.edu>`)

See Also

Other Subpopulation: `P`, `addCloned()`, `addCrossed()`, `addEmpty()`, `addRecombinant()`, `addSelfed()`, `addSpatialMap()`, `cachedFitness()`, `configureDisplay()`, `defineSpatialMap()`, `outputMSSample()`, `outputSample()`, `outputVCFSample()`, `pointDeviated()`, `pointInBounds()`, `pointPeriodic()`, `pointReflected()`, `pointStopped()`, `pointUniform()`, `removeSpatialMap()`, `removeSubpopulation()`, `sampleIndividuals()`, `setCloningRate()`, `setMigrationRates()`,

setSelfingRate(), setSpatialBounds(), setSubpopulationSize(), spatialMapColor(), spatialMapImage(), spatialMapValue(), subsetIndividuals(), takeMigrants()

`setSpatialBounds` *SLiM method setSpatialBounds*

Description

Documentation for SLiM function `setSpatialBounds`, which is a method of the SLiM class `Subpopulation`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
setSpatialBounds(bounds)
```

Arguments

`bounds`	An object of type numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 745.

Set the spatial boundaries of the subpopulation to bounds. This method may be called only for simulations in which continuous space has been enabled with initializeSLiMOptions(). The length of bounds must be double the spatial dimensionality, so that it supplies both minimum and maximum values for each coordinate. More specifically, for a dimensionality of "x", bounds should supply (x0, x1) values; for dimensionality "xy" it should supply (x0, y0, x1, y1) values; and for dimensionality "xyz" it should supply (x0, y0, z0, x1, y1, z1) (in that order). These boundaries will be used by SLiMgui to calibrate the display of the subpopulation, and will be used by methods such as pointInBounds(), pointReflected(), pointStopped(), and pointUniform(). The default spatial boundaries for all subpopulations span the interval [0,1] in each dimension. Spatial dimensions that are periodic (as established with the periodicity parameter to initializeSLiMOptions()) must have a minimum coordinate value of 0.0 (a restriction that allows the handling of periodicity to be somewhat more efficient). The current spatial bounds for the subpopulation may be obtained through the spatialBounds property. The spatial bounds of a subpopulation are shared with any SpatialMap objects added to the subpopulation. For this reason, once a spatial map has been added to a subpopulation, the spatial bounds of the subpopulation can no longer be changed (because it would stretch or shrink the associated spatial map, which does not seem to make physical sense). The bounds for a subpopulation should therefore be configured before any spatial maps are added to it. If those bounds do need to change subsequently, any associated spatial maps must first be removed with removeSpatialMap(), to ensure model consistency.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Subpopulation: P, addCloned(), addCrossed(), addEmpty(), addRecombinant(), addSelfed(), addSpatialMap(), cachedFitness(), configureDisplay(), defineSpatialMap(), outputMSSample(), outputSample(), outputVCFSample(), pointDeviated(), pointInBounds(), pointPeriodic(), pointReflected(), pointStopped(), pointUniform(), removeSpatialMap(), removeSubpopulation(), sampleIndividuals(), setCloningRate(), setMigrationRates(), setSelfingRate(), setSexRatio(), setSubpopulationSize(), spatialMapColor(), spatialMapImage(), spatialMapValue(), subsetIndividuals(), takeMigrants()

---

setSpatialPosition *SLiM method setSpatialPosition*

Description

Documentation for SLiM function setSpatialPosition, which is a method of the SLiM class Individual. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
setSpatialPosition(position)
```

Arguments

position	An object of type float. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 684.

Sets the spatial position of the individual (as accessed through the spatialPosition property). The length of position (the number of coordinates in the spatial position of an individual) depends upon the spatial dimensionality declared with initializeSLiMOptions(). If the spatial dimensionality is zero (as it is by default), it is an error to call this method. The elements of position are set into the values of the x, y, and z properties (if those properties are encompassed by the spatial dimensionality of the simulation). In other words, if the declared dimensionality is "xy", calling individual.setSpatialPosition(c(1.0, 0.5)) property is equivalent to individual.x = 1.0; individual.y = 0.5; individual.z is not set (even if a third value is supplied in position) since it is not encompassed by the simulation's dimensionality in this example. Note that this is an Eidos class method, somewhat unusually, which allows it to work in a special way when called on a vector of individuals. When the target vector of individuals is non-singleton, this method can do one of two things. If position contains just a single point (i.e., is equal in length to the spatial dimensionality of the model), the spatial position of all of the target individuals will be set to the given point. Alternatively, if position contains one point per target individual (i.e., is equal in length to the number of individuals multiplied by the spatial dimensionality of the model), the spatial position of each target individual will be set to the corresponding point from position (where the point data is concatenated, not interleaved, just as it would be returned by accessing the spatialPosition property on the vector of target individuals). Calling this method with a position vector of any other length is an error.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Individual: In, containsMutations(), countOfMutationsOfType(), relatedness(), sharedParentCount(), sumOfMutationsOfType(), uniqueMutationsOfType()

`setSubpopulationSize` *SLiM method setSubpopulationSize*

Description

Documentation for SLiM function `setSubpopulationSize`, which is a method of the SLiM class `Subpopulation`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
setSubpopulationSize(size)
```

Arguments

`size`	An object of type integer. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 745.

Set the size of this subpopulation to size individuals. This will take effect when children are next generated; it does not change the current subpopulation state. Setting a subpopulation to a size of 0 does have some immediate effects that serve to disconnect it from the simulation: the subpopulation is removed from the list of active subpopulations, the subpopulation is removed as a source of migration for all other subpopulations, and the symbol representing the subpopulation is undefined. In this case, the subpopulation itself remains unchanged until children are next generated (at which point it is deallocated), but it is no longer part of the simulation and should not be used.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (`<bhaller@benhaller.com>`) and Philipp W Messer (`<messer@cornell.edu>`)

See Also

Other Subpopulation: `P`, `addCloned()`, `addCrossed()`, `addEmpty()`, `addRecombinant()`, `addSelfed()`, `addSpatialMap()`, `cachedFitness()`, `configureDisplay()`, `defineSpatialMap()`, `outputMSSample()`, `outputSample()`, `outputVCFSample()`, `pointDeviated()`, `pointInBounds()`, `pointPeriodic()`, `pointReflected()`, `pointStopped()`, `pointUniform()`, `removeSpatialMap()`, `removeSubpopulation()`, `sampleIndividuals()`, `setCloningRate()`, `setMigrationRates()`, `setSelfingRate()`, `setSexRatio()`, `setSpatialBounds()`, `spatialMapColor()`, `spatialMapImage()`, `spatialMapValue()`, `subsetIndividuals()`, `takeMigrants()`

`setSuppliedValue` *SLiM method setSuppliedValue*

Description

Documentation for SLiM function `setSuppliedValue`, which is a method of the SLiM class `LogFile`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
setSuppliedValue(columnName, value)
```

Arguments

`columnName`	An object of type string or any but object. Must be of length 1 (a singleton). See details for description.
`value`	An object of type string or any but object. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 703.

Registers a value, passed in value, to be used for the supplied column named columnName when a row is next logged. This column must have been added with addSuppliedColumn(). A value of NULL may be passed to log NA, but logging NA is the default behavior for supplied columns in any case. Otherwise, the value must be a singleton, and its type should match the values previously supplied for the column (otherwise the log file may be difficult to parse, since the values within the column will not be of one consistent type). See addSuppliedColumn() for further details.

Value

An object of type void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or logical.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other LogFile: LF, addCustomColumn(), addCycleStage(), addCycle(), addKeysAndValuesFrom(), addMeanSDColumns(), addPopulationSexRatio(), addPopulationSize(), addSubpopulationSexRatio(), addSubpopulationSize(), addSuppliedColumn(), addTick(), clearKeysAndValues(), flush(), logRow(), setFilePath(), setLogInterval(), setValue(), willAutolog()

---

setValue *SLiM method setValue*

Description

Documentation for SLiM function `setValue`, which is a method of the SLiM class `LogFile`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
setValue(key, value)
```

Arguments

`key`	An object of type string or any. Must be of length 1 (a singleton). See details for description.
`value`	An object of type string or any. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 703.

This Dictionary method has an override in LogFile to make it illegal to call, since LogFile manages its Dictionary entries.

Value

An object of type void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or logical.

Copyright

This is documentation for a function in the SLiM software, and has been reproduced from the official manual, which can be found here: http://benhaller.com/slim/SLiM_Manual.pdf. This documentation is Copyright © 2016-2020 Philipp Messer. All rights reserved. More information about SLiM can be found on the official website: https://messerlab.org/slim/

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other LogFile: LF, addCustomColumn(), addCycleStage(), addCycle(), addKeysAndValuesFrom(), addMeanSDColumns(), addPopulationSexRatio(), addPopulationSize(), addSubpopulationSexRatio(), addSubpopulationSize(), addSuppliedColumn(), addTick(), clearKeysAndValues(), flush(), logRow(), setFilePath(), setLogInterval(), setSuppliedValue(), willAutolog()

SG *SLiMgui*

Description

Documentation for SLiMgui class from SLiM

Details

This class represents the SLiMgui application. When running under SLiMgui, a global object singleton constant of class SLiMgui will be defined, named slimgui. This object can be used to query and control the SLiMgui application. When running at the command line, the slimgui object will not exist; to determine whether the simulation is running under SLiMgui, one may therefore test exists("slimgui"). If a model needs to run both at the command line and under SLiMgui, all uses of the slimgui object should be protected by if (exists("slimgui")) to avoid errors. This class has the following methods (functions):

- openDocument
- pauseExecution

This class has the following properties:

pid A property of type integer. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The Un*x process identifier (commonly called the "pid") of the running SLiMgui application. This can be useful for scripts that wish to use system calls to influence the SLiMgui application.

See Also

Other SLiMgui: `openDocument()`, `pauseExecution()`

`sharedParentCount` *SLiM method sharedParentCount*

Description

Documentation for SLiM function `sharedParentCount`, which is a method of the SLiM class `Individual`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
sharedParentCount(individuals)
```

Arguments

`individuals`	An object of type Individual object. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 684.

Returns a vector containing the number of parents shared between the receiver and each of the individuals in individuals. The number of shared parents between A and B is always 2 if A and B are actually the same individual; this facility works even if SLiM's optional pedigree tracking is not enabled (in which case all other relatedness values will be 0). Otherwise, if pedigree tracking is turned on with initializeSLiMOptions(keepPedigrees=T), this method will use the pedigree information described in section 25.7.1 to construct a relatedness estimate. More specifically, this method uses the parental pedigree IDs from the pedigree records of a pair of individuals to count the number of shared parents between them, such that full siblings (with all of the same parents) have a count of 2, and half siblings (with half of the same parents) have a count of 1. If possible parents of the two individuals are A, B, C, and D, then the shared parent count is as follows, for some illustrative examples. The first column showing the two parents of the first individual, the second column showing the two parents of the second individual; note that the two parents of an individual can be the same due to cloning or selfing: AB CD → 0 (no shared parents) AB CC → 0 (no shared parents) AB AC → 1 (half siblings) AB AA → 1 (half siblings) AA AB → 1 (half siblings) AB AB → 2 (full siblings) AB BA → 2 (full siblings) AA AA → 2 (full siblings) This method does not estimate consanguinity. For example, if one individual is itself a parent of the other individual, that is irrelevant for this method. Similarly, in simulations of sex chromosomes, the sexes of the parents are irrelevant, even if no genetic material would have been inherited from a given parent. See relatedness() for an assessment of pedigree-based relatedness that does estimate the consanguinity of individuals. The sharedParentCount() method is preferable if your exact question is simply whether individuals are full siblings, half siblings, or non-siblings; in other cases, relatedness() is probably more useful.

Value

An object of type integer.

Copyright

This is documentation for a function in the SLiM software, and has been reproduced from the official manual, which can be found here: http://benhaller.com/slim/SLiM_Manual.pdf. This documentation is Copyright © 2016-2020 Philipp Messer. All rights reserved. More information about SLiM can be found on the official website: https://messerlab.org/slim/

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Individual: In, containsMutations(), countOfMutationsOfType(), relatedness(), setSpatialPosition(), sumOfMutationsOfType(), uniqueMutationsOfType()

---

simulationFinished *SLiM method simulationFinished*

Description

Documentation for SLiM function simulationFinished, which is a method of the SLiM class Community. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Documentation for SLiM function simulationFinished, which is a method of the SLiM class Species. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
simulationFinished(void)

simulationFinished(void)
```

Arguments

void	An object of type . See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 668.

Declare the current simulation finished. Normally SLiM ends a simulation when, at the end of a tick, there are no script events or callbacks registered for any future tick (excluding scripts with no declared end tick). If you wish to end a simulation before this condition is met, a call to simulationFinished() will cause the current simulation to end at the end of the current tick. For example, a simulation might self-terminate if a test for a dynamic equilibrium condition is satisfied. Note that the current tick will finish executing; if you want the simulation to stop immediately, you can use the Eidos method stop(), which raises an error condition.

Documentation for this function can be found in the official SLiM manual: page 727.

Declare the current simulation finished. This method is equivalent to the Community method simulationFinished(), except that this method is only legal to call in single-species models (to provide backward compatibility). It is recommended that new code should call the Community method; this method may be deprecated in the future.

Value

An object of type void.

An object of type void.

Copyright

This is documentation for a function in the SLiM software, and has been reproduced from the official manual, which can be found here: http://benhaller.com/slim/SLiM_Manual.pdf. This documentation is Copyright © 2016-2020 Philipp Messer. All rights reserved. More information about SLiM can be found on the official website: https://messerlab.org/slim/

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Community: Co, createLogFile(), deregisterScriptBlock(), genomicElementTypesWithIDs(), interactionTypesWithIDs(), mutationTypesWithIDs(), outputUsage(), registerEarlyEvent(), registerFirstEvent(), registerInteractionCallback(), registerLateEvent(), rescheduleScriptBlock(), scriptBlocksWithIDs(), speciesWithIDs(), subpopulationsWithIDs(), usage()

Other Species: Sp, addSubpopSplit(), addSubpop(), countOfMutationsOfType(), individualsWithPedigreeID killIndividuals(), mutationCounts(), mutationFrequencies(), mutationsOfType(), outputFixedMutations(), outputFull(), outputMutations(), readFromPopulationFile(),

`recalculateFitness()`, `registerFitnessEffectCallback()`, `registerMateChoiceCallback()`, `registerModifyChildCallback()`, `registerMutationCallback()`, `registerMutationEffectCallback()`, `registerRecombinationCallback()`, `registerReproductionCallback()`, `registerSurvivalCallback()`, `skipTick()`, `subsetMutations()`, `treeSeqCoalesced()`, `treeSeqOutput()`, `treeSeqRememberIndividuals()`, `treeSeqSimplify()`

`skipTick` *SLiM method skipTick*

Description

Documentation for SLiM function `skipTick`, which is a method of the SLiM class `Species`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
skipTick(void)
```

Arguments

`void`	An object of type . See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 727.

Deactivate the target species for the current tick. This sets the active property of the species to F; it also set the active property of all callbacks that belong to the species (with the species as their species specifier) to F, and sets the active property of all events that are synchronized with the species (with the species as their ticks specifier) to F. The cycle counter for the species will not be incremented at the end of the tick. This method may only be called in first() events, to ensure that species are either active or inactive throughout a given tick.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Species: Sp, addSubpopSplit(), addSubpop(), countOfMutationsOfType(), individualsWithPedigreeID
killIndividuals(), mutationCounts(), mutationFrequencies(), mutationsOfType(),
outputFixedMutations(), outputFull(), outputMutations(), readFromPopulationFile(),
recalculateFitness(), registerFitnessEffectCallback(), registerMateChoiceCallback(),
registerModifyChildCallback(), registerMutationCallback(), registerMutationEffectCallback(),
registerRecombinationCallback(), registerReproductionCallback(), registerSurvivalCallback(),
simulationFinished(), subsetMutations(), treeSeqCoalesced(), treeSeqOutput(),
treeSeqRememberIndividuals(), treeSeqSimplify()

---

slimr_clip_original *Copy original slimrlang code used to create a slimr_script object to the clipboard*

Description

Copy original slimrlang code used to create a slimr_script object to the clipboard

Usage

```
slimr_clip_original(slimr_script_name = NULL)
```

Arguments

`slimr_script_name`
: Name of slimr_script to copy its original code to the clipboard, as a string

Value

The code as a string (invisibly)

Examples

```
test <- minimal_slimr_script()
if(rlang::is_installed("clipr") && interactive()) {
    slimr_clip_original("test")
}
```

slimr_code *Create a* slimr_code *object*

Description

Create a slimr_code object

Usage

```
slimr_code(...)
```

Arguments

...	A list of character vectors.

Examples

```
slimr_code('print("Hello")', 'print("World!")')
```

slimr_name *Convert a string to a SLiM object name*

Description

Convert a string to a SLiM object name

Usage

```
slimr_name(name)
```

Arguments

name	A string with the SLiM object name

Value

A symbol

Examples

```
slimr_name("p1")
```

slimr_open_original *Open original slimrlang code used to create a slimr_script object in a new document*

Description

Open the original code used to create a slimr_script in a new document. Requires RStudio.

Usage

```
slimr_open_original(slimr_script_name = NULL)
```

Arguments

`slimr_script_name`
: Name of slimr_script to open in a new document as a string.

Value

The code as a string (invisibly)

Examples

```
test <- minimal_slimr_script()
if(rlang::is_installed("rstudioapi") && interactive()) {
    slimr_open_original("test")
}
```

slimr_script_coll *Make a new* `slimr_script_coll` *object*

Description

Make a new `slimr_script_coll` object

Usage

```
slimr_script_coll(...)
```

Arguments

`...`
: A list of `slir_script` objects

Value

A `slimr_script_coll` object

`slimr_write` *Write a slimr_script object to a text file*

Description

Write a slimr_script object to a text file, which can be run in SLiM as a SLiM script

Usage

```
slimr_write(x, file, ...)
```

Arguments

`x`	`slimr_script` object to write to file
`file`	File path to write to
`...`	Further arguments to be passed to or from other objects

Value

Returns `x`, invisibly

Examples

```
slim_script(
   slim_block(initialize(),
          {
            initializeMutationRate(1e-7);
            initializeMutationType("m1", 0.5, "f", 0.0);
            initializeGenomicElementType("g1", m1, 1.0);
            initializeGenomicElement(g1, 0, 99999);
            initializeRecombinationRate(1e-8);
          }),
   slim_block(1,
          {
            sim.addSubpop("p1", 500);
          }),
   slim_block(10000,
          {
            sim.simulationFinished();
          })
) -> script
temp_file <- tempfile(fileext = ".txt")
slimr_write(script, temp_file)
readLines(temp_file)
```

`slim_block` *Setup a SLiM code block*

Description

`slim_block` sets up and Eidos event code block. See details for how to specify the arguments correctly.

Usage

```
slim_block(...)
```

Arguments

`...`	A list of arguments corresponding to elements in SLiM code blocks. See details for more information on how to specify these arguments.

Details

An Eidos event is a block of Eidos code that is executed every generation, within a generation range, to perform a desired task. The syntax of an Eidos event declaration in `slimr` mimics that of the Eidos (e.g. SLiM) language itself (see SLiM manual). It looks like this: `slim_block([species_id = ][id,] [start_gen, [end_gen,]], [slim_callback,] { ... })` where [ ] specifies that the code is optional. The minimum required is a single argument containing Eidos code. This will be run in every generation with slim_callback `early()`, the default for Eidos events. You can also optionally specify an id for the SLiM code block, which will be the first argument. This is optionally followed by a starting generation (start_gen). If only a starting generation is specified, the event will run only in that generation. Next comes an optional end generation (end_gen), which, if specified, will tell SLiM to run the event every generation between start_gen and end_gen. The special value of `..` can be used for end_gen instead, which is shorthand for the last generation (in other words, run the event every generation between start_gen and the last generation used else where in the script). After end_gen is an optional callback, which corresponds to an Eidos callback. The following are valid Eidos callbacks:

- `early()`
- `late()`
- `initialize()`
- `fitness(mut_type_id, subpop_id)`
- `mateChoice(subpop_id)`
- `modifyChild(subpop_id)`
- `recombination(subpop_id)`
- `interaction(int_type_id, subpop_id)`
- `reproduction(subpop_id, sex)`

Multispecies models are supported (if using SLiM >= 4.0), by using a single named argument where the argument name is a species id (e.g. `slim_block(species_id = early())` would create a block that would run early in every generation and apply only to species `species_id`). Note that any of the arguments can be named, but only one. It is usually easiest to name the first argument specified in `slim_block()`. This may seem a somewhat unusual way to specify a species id but it was the simplest way to support multispecies models without changing the syntax of `slim_block()` much and maintaining the conciseness of block declarations which is a hallmark of SLiM and `slimr`.

Value

A slimr_block object. This is of little use outside a `slim_script` function call.

Examples

```
slim_script(slim_block("s1", 1, 10000, late(), {print("Hello World!")}))
```

---

slim_block_add_subpops

Generate a code block that just adds subpopulations to a SLiM simulation

---

Description

Generate a code block that just adds subpopulations to a SLiM simulation

Usage

```
slim_block_add_subpops(
  n_pop = 1,
  sizes,
  generation = 1,
  when = c("early", "late")
)
```

Arguments

`n_pop`	Number of subpopulations to add
`sizes`	Population sizes of subpopulations. Should have a length equal to `n_pop`, or else a length of 1 (in which case it will be recycled to `n_pop`)
`generation`	The generation to add the subpopulations (default: 1)
`when`	When to add the subpopulations ("early" or "late"). Default: "early"

Value

A `slimr_block` object

Examples

```
slim_block_add_subpops(2, 100)
```

`slim_block_finish` *Generate simulationFinished Block*

Description

This generates a simple block to end a SLiM simulation.

Usage

```
slim_block_finish(generation)
```

Arguments

`generation`	Generation to end the simulation at.

Value

A `slimr_block` object

Examples

```
slim_script(
  slim_block_init_minimal(),
  slim_block_finish(100)
)
```

`slim_block_init_minimal`
Generate minimal initialize() block

Description

This function creates a minimal initialize block, allowing you to set some basic parameters, but generally only allowing for one type of mutation, which is distributed across the whole genome.

Usage

```
slim_block_init_minimal(
  mutation_rate = 1e-07,
  dominance = 0.5,
  selection = 0,
  dist_type = "f",
  genome_size = 1e+05,
  recombination_rate = 1e-08,
  seed = NULL
)
```

Arguments

`mutation_rate`	The overall mutation rate.
`dominance`	The overall dominance value.
`selection`	Mean selection strength for mutations
`dist_type`	Distribution from which to draw mutation selection values (see `initializeMutationType` for possible values).
`genome_size`	Genome size of the population, in number of loci.
`recombination_rate`	Overall recombination rate.
`seed`	An optional integer used to set a random seed for the SLiM simulation.

Value

A `slimr_block` object

Examples

```
slim_script(
  slim_block_init_minimal(),
  slim_block_finish(100)
)
```

`slim_block_progress` *Insert slim_block to track progress when using slim_run*

Description

Insert slim_block to track progress when using slim_run

Usage

```
slim_block_progress(update_every = 1, time_counter = community.tick)
```

Arguments

update_every	How often to update progress, expressed as the number of generations after which an update should be tracked.

Value

A slimr_block object

Examples

```
slim_script(
  slim_block_init_minimal(),
  slim_block_progress(10),
  slim_block_finish(100)
)
```

---

slim_callbacks *SLiM Callbacks*

Description

These functions are stubs that don't do anything in R, but represent callback functions in SLiM. Call these functions as part of a slim_block call to specify that the code is designed to work with a particular callback. Most callbacks are designed to be called within SLiM under special circumstances, and will have 'pseudo-variables' that are accessible only inside that particular callback. See help files for individuals callbacks listed below for more details. Available callbacks are:

- initialize
- early
- late
- fitness
- fitnessEffect
- mateChoice
- modifyChild
- mutation
- mutationEffect
- recombination
- interaction
- reproduction
- survival

Usage

```
slim_callbacks()
```

See Also

Other callbacks: `early()`, `first()`, `fitnessEffect()`, `fitness()`, `initialize()`, `interaction()`, `late()`, `mateChoice()`, `modifyChild()`, `mutationEffect()`, `mutation()`, `recombination()`, `reproduction()`, `survival()`

`slim_classes` *SLiM Classes*

Description

A data.frame containing the name of the SLiM classes and an abbreviation that can be used to refer to them in R.

Usage

```
slim_classes
```

Format

A data frame with 15 rows and 2 variables:

class_name The name of SLiM class

class_internal The name used internally by 'slimr' for the class. Generally not useful for users.

class_abbr An abbreviation of the class name that can be used to save typing.

`slim_code_Rify` *Rify some SLiM code*

Description

Utility code to convert SLiM code into a form that can be parsed by R (e.g. in `styler` or `prettycode`). Don't forget to re-SLiMify afterwards (via `slim_code_SLiMify`)!

Usage

```
slim_code_Rify(code_snippet)
```

Arguments

`code_snippet` SLiM code to Rify as a character vector

Value

Rified code snippet

slim_code_SLiMify *Rify some SLiM code*

Description

Utility code to convert Rified (via `slim_code_Rify`) SLiM code back to valid SLiM code!

Usage

```
slim_code_SLiMify(code_snippet)
```

Arguments

`code_snippet`	SLiM code to Rify as a character vector

Value

SLiMified code snippet

slim_extract_full *Extract Elements from SLiM's outputFull()*

Description

Extract Elements from SLiM's outputFull()

Usage

```
slim_extract_full(
  output_full,
  type = c("mutations", "individuals", "genomes", "coordinates", "sexes", "ages",
    "full_individual"),
  join = TRUE,
  expand_mutations = FALSE
)
```

Arguments

`output_full`	A character vector where each element is the result of a call to `outputFull()` in SLiM
`type`	Which type of data to return: "mutations", "individuals", "genomes", "coordinates", "sexes", or "ages"
`join`	If asking for multiple output type, should they be joined into one tibble (`join = TRUE`) or left as separate tibbles returned in a list (`join = FALSE`)?
`expand_mutations`	If asking for "genomes" output, should mutations be expanded into their own column (`expand_mutations = TRUE`) or left as a vector of mutation ids in a list column (`expand_mutations = FALSE`)?

Value

A tibble

Examples

```
if(slim_is_avail()) {
  test_sim <- slim_script(
    slim_block_init_minimal(mutation_rate = 1e-6),
    slim_block_add_subpops(1, 100),
    slim_block(1, 20, late(), {
      r_output(sim.outputFull(), "out", do_every = 10)
    })
  ) %>%
    slim_run()
  slim_extract_full(test_sim$output_data, type = "mutations")
}
```

---

slim_extract_genlight

Extract data into a genlight object

Description

Extract data into a genlight object

Usage

```
slim_extract_genlight(x, ...)
```

Arguments

x	`slimr_results` object containing data generated by `r_output` using `outputFull()` in SLiM
...	Arguments passed to or from other methods.

Value

A `genlight` object

Examples

```
if(slim_is_avail()) {
  test_sim <- slim_script(
    slim_block_init_minimal(mutation_rate = 1e-6),
    slim_block_add_subpops(1, 100),
    slim_block(1, 20, late(), {
      r_output(sim.outputFull(), "out", do_every = 10)
    })
  ) %>%
```

```
  slim_run()
if(require("adegenet", quietly = TRUE)) {
  plot(slim_extract_genlight(test_sim$output_data))
}
}
```

`slim_extract_genome` *Extract Elements from SLiM's output() for genomes*

Description

Extract Elements from SLiM's output() for genomes

Usage

```
slim_extract_genome(
  output,
  type = c("mutations", "genomes", "full"),
  join = TRUE,
  expand_mutations = FALSE
)
```

Arguments

`output`	A character vector where each element is the result of a call to `genomes.output()` in SLiM
`type`	Which type of data to return: "mutations", or "genomes" or both.
`join`	If asking for multiple output type, should they be joined into one tibble (`join = TRUE`) or left as separate tibbles returned in a list (`join = FALSE`)?
`expand_mutations`	If asking for "genomes" output, should mutations be expanded into their own column (`expand_mutations = TRUE`) or left as a vector of mutation ids in a list column (`expand_mutations = FALSE`)?

Value

A tibble

Examples

```
if(slim_is_avail()) {
  test_sim <- slim_script(
    slim_block_init_minimal(mutation_rate = 1e-6),
    slim_block_add_subpops(1, 100),
    slim_block(1, 20, late(), {
      r_output(p1.genomes.output(), "out", do_every = 10)
    })
  ) %>%
```

```
    slim_run()
  slim_extract_genome(test_sim$output_data, type = "mutations")
}
```

`slim_extract_output_data` *Extract data produced by SLiM*

Description

This function takes output produced from a `slimr_script` which uses `r_output`, and converts it into a `tibble`

Usage

```
slim_extract_output_data(output)
```

Arguments

`output`	Character vector produced from SLiM run.

Value

A `tibble` with four columns:

generation A vector of generations where output was produced.

name Names of the output data.

expression The SLiM expression used to generate the output.

data The raw data output from SLiM as a character vector.

Examples

```
if(slim_is_avail()) {
  test_sim <- slim_script(
    slim_block_init_minimal(),
    slim_block_add_subpops(1, 100),
    slim_block(1, 20, late(), {
      r_output(p1$size(), "MS", do_every = 10)
    })
  ) %>%
    slim_run(simple = TRUE)
  slim_extract_output_data(test_sim$output)
}
```

slim_file — *Make sure a file name is compatible with using in a SLiM script.*

Description

This generally only need to be used if you are using Windows, where this function will convert the file name into a path that can be accessed inside Windows Subsystem for Linux (WSL).

Usage

```
slim_file(file_name)
```

Arguments

file_name	The name of the file to convert

Value

The converted filename that will ensure SLiM can access the file.

slim_function — *Specify an Eidos function to be included in the SLiM script*

Description

Specify an Eidos function to be included in the SLiM script

Usage

```
slim_function(..., name, return_type = "f$", body)
```

Arguments

...	List of arguments specified using Eidos's argument syntax (which includes type specification; see details)
name	Name of function being created.
return_type	Type of the functions return, using Eidos' type syntax (see details)
body	SLiM / Eidos code to be executed in the body of the function. Can also be an R function, in which case the body of the R function is used.

Value

A slimr_block object (only useful with the context of a slim_script) call.

slim_get_recipe *Get a SLiM recipe*

Description

Get a SLiM recipe

Usage

```
slim_get_recipe(recipe = 1)
```

Arguments

recipe	Integer or Character specifying the recipe number, or the recipe name.

Value

A SLiM recipe as a length 1 character vector

Examples

```
slim_get_recipe(recipe = "4.1")
```

slim_get_recipes *Get a list of SLiM recipes (as text)*

Description

Get a list of SLiM recipes (as text)

Usage

```
slim_get_recipes(recipes = "all")
```

Arguments

recipes	Integer or Character specifying the recipe numbers / names, or ”all”, to retrieve all recipes.

Value

List of SLiM recipes, each element a length 1 character vector.

Examples

```
recipes <- slim_get_recipes("all")
recipes[[12]]
```

slim_install_path *Find the SLiM executable path 'slimr' is using*

Description

Returns the SLiM installation path used by 'slimr' to run simulations.

Usage

```
slim_install_path()
```

Examples

```
slim_install_path()
```

slim_is_avail *Check if SLiM is installed and* slimr *can find it*

Description

Check if SLiM is installed and `slimr` can find it

Usage

```
slim_is_avail()
```

Value

`TRUE` if SLiM is found and `FALSE` otherwise

Examples

```
slim_is_avail()
```

`slim_load_globals` *Load SLiM Globals into R Global environment*

Description

This function loads R objects to stand in for standard Globals used in SLiM. These will be available in the global environment and can be used to aid in autocompletion and help retrieval, saving typing of this pattern: `p1%.%Subpopulation$setSubpopulationSize(N)` – instead one can just type: `p1$setSubpopulationSize(N)`: this will be replaced by correct SLiM code: `p1.setSubpopulationSize(N)` By default this will load ten of each of the standard SLiM Globals (e.g. objects prefixed with "p", "m", "i", "g", and "s"), as well as the "sim" and "self" singular Globals. Note: be careful with this function. If you have any objects in your R session's global environment with these names they will be overwritten.

Usage

```
slim_load_globals(
  max = 10,
  sim = TRUE,
  community = TRUE,
  self = TRUE,
  pseudo = TRUE
)
```

Arguments

`max`	A single integer values indicating how many of each numbered Globals to load, or a named integer vector where the names refer to different Global types (see details for more information), and the value refers to how many of that Global type to load.
`sim`	Should the `sim` global be loaded?
`community`	Should the `community` global be loaded?
`self`	Should the `self` global be loaded?
`pseudo`	Should 'pseudo-variables' be loaded? These are similar to SLiM globals but are only available inside certain callbacks.

Value

None

Examples

```
slim_load_globals(c(p = 4, g = 2))
```

`slim_make_pop_input` *Create a file to initialise a population in SLiM*

Description

Make a SLiM file in the style of `outputFull()` which can be read by SLiM. This is useful to start a simulation with a particular population state that you define in R

Usage

```
slim_make_pop_input(
  snps,
  file_name = tempfile(),
  sim_gen = 10000,
  ind_pops = NULL,
  ind_sex = NULL,
  mut_pos = NULL,
  mut_type = NULL,
  mut_sel = NULL,
  mut_dom = NULL,
  mut_pop = NULL,
  mut_gen = NULL,
  mut_nuc = NULL,
  version = 4
)
```

Arguments

`snps`	R object containing SNPs. Can be a matrix with 0, 1, or 2 as its elements, or a `genlight` object. Rows correspond to individuals, columns to loci.
`file_name`	Path to file where the population input file should be saved.
`sim_gen`	What generation to write in the file. This doesn't really do anything, but is just a requirement for the format.
`ind_pops`	An optional character vector with length equal to `nrow(snps)` with Subpopulation indicators for each individual. Should be "p1", "p2", etc. as this is how subpopulations are named in SLiM.
`ind_sex`	An optional character vector with length equal to `nrow(snps)` with the sex for each individual (can be "H" for hermaphrodite, "F" for female, or "M" for male.)
`mut_pos`	An optional integer vector with length equal to `ncol(snps)` specifying the positions in the genome of each loci.
`mut_type`	An optional character vector with length equal to `nrow(snps)` with the mutation type of each loci. Should be "m1", "m2", etc. as this is how mutation types are named in SLiM.

`mut_sel`	An optional numeric vector with length equal to `nrow(snps)` with the selection coefficient for the mutation type of each loci.
`mut_dom`	An optional numeric vector with length equal to `nrow(snps)` with the dominance coefficient for the mutation type of each loci.
`mut_pop`	An optional character vector with length equal to `nrow(snps)` with the subpopulation of origin for the mutation at each loci. Should be "p1", "p2", etc. as this is how subpopulations are named in SLiM.
`mut_gen`	An optional integer vector with length equal to `ncol(snps)` specifying the generation of origin for the mutation at each loci.
`mut_nuc`	An optional character vector with length equal to `nrow(snps)` with the nucleotide of the mutation. Should be "A", "C", "G", or "T".
`version`	SLiM output version number to use.

Value

Returns the file name where the population data was saved

Examples

```
pop_file <- slim_make_pop_input(matrix(rbinom(1000, 2, 0.25), nrow = 100, ncol = 100))
cat(readLines(pop_file))
```

`slim_open` *Title*

Description

Title

Usage

```
slim_open(slimr_script, slim_gui_path = Sys.getenv("slim_gui_path"))
```

Arguments

`slimr_script`	A slimr_script object to open in SLiMGUI (or QtSLiM if on Linux or Windows)
`slim_gui_path`	Full path to SLiMGUI or QtSLiM executable

Value

Invisibly return the process object used to launch the GUI (for debugging purposes)

Examples

```
if(interactive()) {
  slim_open(minimal_slimr_script())
}
```

slim_recipes *Dataset of all recipes available in SLiM*

Description

A Dataset containing all SLiM recipes found in the SLiM Manual

Usage

```
slim_recipes
```

Format

A List with 190 elements, each containing a character vector

Source

https://github.com/MesserLab/SLiM/releases/download/v4.1/SLiM_Manual.pdf

slim_results_to_data *Convert output data from a slimr_results object to a tibble*

Description

This function tries to automate the process of converting output in the `slimr_results` object returned by `slim_run` function into usable data in the form of a `tibble`. If this process fails you will end up with the data as a character string, and you will have to manually convert this into something you can use.

Usage

```
slim_results_to_data(dat, generations = NULL)
```

Arguments

`dat`	A `slimr_results` object to extract data from. You can alternatively specify the `output_data` from the `slimr_results` directly in this parameter.
`generations`	For what generations do you want to extract data? Default is all generations that have data. The special value `0` can be used to specify just the most recent generation in the data.

Value

A tibble with three columns, name: the name of the outputs, generation: the generations of the outputs, and data: the outputs as converted data. This final column will be a list column, where each element will usually be a vector or a tibble, depending on what kind of data was returned by the simulation.

Examples

```
if(slim_is_avail()) {
  test_sim <- slim_script(
    slim_block_init_minimal(),
    slim_block_add_subpops(1, 100),
    slim_block(1, 20, late(), {
      r_output(sim.outputFull(), "out", do_every = 10)
    })
  ) %>%
    slim_run()
  slim_results_to_data(test_sim)
}
```

---

slim_run *Run a SLiM script from R*

Description

This function runs a SLiM script, specified as a `slimr_script` object, a character vector, or a text file.

Usage

```
slim_run(
  x,
  slim_path = NULL,
  script_file = NULL,
  simple_run = FALSE,
  capture_output = "file",
  keep_all_output = FALSE,
  show_output = FALSE,
  callbacks = NULL,
  cb_args = NULL,
  new_grdev = FALSE,
  record_graphics = "",
  rec_args = NULL,
  parallel = FALSE,
  progress = FALSE,
  throw_error = FALSE,
  ...
)

## S3 method for class 'character'
slim_run(
  x,
  slim_path = NULL,
  script_file = NULL,
```

```
  simple_run = FALSE,
  capture_output = "file",
  keep_all_output = FALSE,
  show_output = FALSE,
  callbacks = NULL,
  cb_args = NULL,
  new_grdev = FALSE,
  record_graphics = "",
  rec_args = NULL,
  parallel = FALSE,
  progress = FALSE,
  throw_error = FALSE,
  ...
)

## S3 method for class 'slimr_script'
slim_run(
  x,
  slim_path = NULL,
  script_file = NULL,
  simple_run = FALSE,
  capture_output = "file",
  keep_all_output = FALSE,
  show_output = FALSE,
  callbacks = NULL,
  cb_args = NULL,
  new_grdev = FALSE,
  record_graphics = "",
  rec_args = NULL,
  parallel = FALSE,
  progress = FALSE,
  throw_error = FALSE,
  ...
)

## S3 method for class 'slimr_script_coll'
slim_run(
  x,
  slim_path = NULL,
  script_file = NULL,
  simple_run = FALSE,
  capture_output = "file",
  keep_all_output = FALSE,
  show_output = FALSE,
  callbacks = NULL,
  cb_args = NULL,
  new_grdev = FALSE,
  record_graphics = "",
```

```
  rec_args = NULL,
  parallel = FALSE,
  progress = FALSE,
  throw_error = FALSE,
  ...
)
```

Arguments

`x`	Object containing script to run (e.g. a character vector or a slimr_script object)
`slim_path`	Path to the SLiM executable. If left `NULL` `slimr` will attempt to automatically determine it, typically by examining environmental variables.
`script_file`	If the script you want to run is in a text file, you can add the pather here. If this is argument is not `NULL` argument `x` will be ignored
`simple_run`	Whether to do a "simple run", which just runs the script, capturing all output is `capture_output` is `TRUE` and additionally sending all output to the R console if `show_output` is `TRUE` the script to the R console if show_output is TRUE
`capture_output`	If `TRUE`, output from the script will be captured and included in the returned object. Unless `keep_all_output` is `TRUE`, only non-data output will be kept (e.g. output not produced by a `r_output` call)
`keep_all_output`	If there is data produced by `r_output` calls, should it be captured as well? Ignored if `capture_output` is not `TRUE`
`show_output`	Should output from the script be sent to the R console? Note that SLiM scripts can sometimes produce a large amount of output, which could overwhelm the console if you are not careful, potentially locking it up. Be careful with this option if you are using any of SLiM's output functions that output genomic data. This can be handy though for simply status print outs..
`callbacks`	A list of functions to be called during the SLiM run. This can be used to dynamically transform or visualise output from the simulation while it is running. It should be of the form `function(data, ...) {do something..}`. If using `r_output` to get formatted data output from SLiM, data will be a four column `tibble` containing output from the current iteration of the simulation. Columns are: **generation** A vector of generations processed in the current iteration **name** Names of the output data. **expression** The SLiM expression used to generate the output **data** The raw data output from SLiM as a character vector
`cb_args`	Additional arguments to be passed to any callback functions. Should be a named list where the names refer to the callback's arguments.
`new_grdev`	Should a new graphics device window be opened on RStudio? This is mainly useful if you are using custom callbacks that generate live figures,

and want a faster plotting experience. This is because the default plot viewer in RStudio can be quite slow. Setting this to `TRUE` also allows `record_graphics` to work.

`record_graphics`
: An optional character string specifying a file to record video output from the graphics device. If you have custom graphics output for the simulation this lets you record that output live, saving the trouble of producing separate animations after the simulation is complete (also allowing you to save memory because you don't need to keep the entire simulation output to make a post-hoc animation).

`rec_args`
: An optional list containing named arguments to be passed to `av_capture_graphics` for graphics recording. Ignored if `record_graphics` is not `TRUE`.

`parallel`
: If `x` is a `slimr_script_coll`, should the elements in `x` be run in parallel. For this to work, you must have setup a parallel plan using `plan`

`progress`
: Should a progress bar be displayed?

`throw_error`
: Should an error be thrown in R is an error is encountered in SLiM? If `FALSE`, the error message from SLiM is stored in the object returned by `slim_run`, but execution continues in R. Setting this to `TRUE` is useful in a script if subsequent code assumes that the simulation finished successfully.

`...`
: Additional arguments to be passed to or from other methods.

Value

A `slimr_results` object which has the following components:

output A character vector of raw output. Will be NULL if `capture_output` is `FALSE`

exit_status The exit status code returned by the SLiM process. 0 means success.

output_data A 'tibble' containing output from `r_output` calls.

process A `processx` object containing information about the SLiM process used during the run.

error If an error was encountered during the run, this will be a character vector containing the error message.

output_file The path to the file containing captured output from SLiM during the run.

Methods (by class)

- `slim_run(character)`: Run a SLiM script from character vector
- `slim_run(slimr_script)`: Run a SLiM script from slimr_script object
- `slim_run(slimr_script_coll)`: Run a SLiM script from slimr_script object

Examples

```
if(slim_is_avail()) {
  test_sim <- slim_script(
    slim_block_init_minimal(mutation_rate = 1e-6),
    slim_block_add_subpops(1, 100),
```

```
    slim_block(1, 20, late(), {
      r_output(sim.outputFull(), "out", do_every = 10)
    })
  ) %>%
    slim_run()
  test_sim
}
```

`slim_script` *Create a SLiMR script*

Description

Setup a SLiMR script. Each argument should be a call to `slim_block`. See details for more information.

Usage

```
slim_script(...)
```

Arguments

`...`	A list of `slim_block` objects comprising a SLiM script (written in slimr code)

Value

A `slim_script` object that can be used with `slim_run` or converted into a text file for use with SLiM directly using `as_slim_text`.

Examples

```
slim_script(
   slim_block(initialize(),
          {
            initializeMutationRate(1e-7);
            initializeMutationType("m1", 0.5, "f", 0.0);
            initializeGenomicElementType("g1", m1, 1.0);
            initializeGenomicElement(g1, 0, 99999);
            initializeRecombinationRate(1e-8);
          }),
   slim_block(1,
          {
            sim.addSubpop("p1", 500);
          }),
   slim_block(10000,
          {
            sim.simulationFinished();
          })
)
```

slim_script_duration *Set the duration of a slimr_script*

Description

Set the duration of a slimr_script

Usage

```
slim_script_duration(x, duration)
```

Arguments

x	slimr_script object to set the duration of
duration	Final generation number to run for this simulation

Value

A `slimr_script` object with new duration

Examples

```
test_sim <- slim_script(
  slim_block_init_minimal(mutation_rate = 1e-6),
  slim_block_add_subpops(1, 100),
  slim_block(1, 20, late(), {
    slimr_output(sim.outputFull(), "out", do_every = 10)
  })
)

slim_script_duration(test_sim, 100)
```

slim_script_render *Render a SLiM script with special slimrlang formatting*

Description

If your `slimr_script` object has made use of special `slimrlang` syntax `slimr_template`, this function will 'render' the `slimr_script` into valid SLiM syntax, ready to be run with SLiM or `slim_run`

Usage

```
slim_script_render(
  slimr_script,
  template = NULL,
  replace_NAs = TRUE,
  reps = 1,
  parallel = FALSE,
  portable = FALSE
)
```

Arguments

slimr_script	The slimr_script object to be rendered
template	A list or data.frame containing values for any templated variables. If a list, it must be named, where the names correspond to the variables. If a list of lists, the inner lists must be named with the variable names, and slim_script_render will render a separate slimr_script for each top-level list element and return it as a slimr_script_coll object. If a data.frame (or tibble), then the column names should match the templated variables, and slim_script_render will render a separate slimr_script for each row and return it as a slimr_script_coll object.
replace_NAs	Should NA values in the template be replaced by their default values?
reps	Should the rendered script be replicated? If greater than 1, a slimr_script_coll will be returned. This can also be used with a slimr_script object that has already been rendered, in which case it will just repeat the rendered script in the result.
parallel	Should the rendering be done in parallel when rendering multiple scripts? Requires the furrr package and will use the plan set by future::plan
portable	If 'TRUE', the the script will be rendered in a 'portable' format, which allows the script to be modified in another program such as SLiMGUI, and then reimported into R, while maintaining 'slimr' features. See details for more information on how this works.

Value

A rendered 'slimr_script_coll' object

Examples

```
test_sim <- slim_script(
  slim_block_init_minimal(mutation_rate = r_template("mu", 1e-7)),
  slim_block_add_subpops(1, 100),
  slim_block(1, 20, late(), {
    r_output(sim.outputFull(), "out", do_every = 10)
  })
) %>%
  slim_script_render(data.frame(mu = c(1e-7, 1e-6, 1e-5)))
test_sim
```

`slim_setup` *Attempt to install and / or setup SLiM for use with slimr*

Description

'slim_setup()' will attempt to determine the user's OS and install SLiM automatically. Note that on Windows, this will attempt to download a precompiled executable.

Usage

```
slim_setup(
  method = c("conda", "binary"),
  verbose = TRUE,
  force = FALSE,
  install_path = default_install_path(),
  conda_env = "slimr-conda"
)
```

Arguments

`method`	The method to use to install SLiM. Currently, only "conda" and "binary" are supported. If "conda", then SLiM will be installed via conda using the `reticulate` package. If "binary", then SLiM will be downloaded as a precompiled executable, which is currently only available on Windows.
`verbose`	Whether to print out progress of the installation.
`force`	If `FALSE` (the default) `slim_setup` will not install SLiM if it is already installed and can be found. If you want to force an installation, even if SLiM is already installed (perhaps to install a newer version), then use `force=TRUE`.
`install_path`	If `method="binary"`, then this is the path to install the SLiM executable. If you do not use the default path, then you will need to set the `SLIM_PATH` environment variable so that slimr can find it.
`conda_env`	If `method="conda"`, then this is the name of the conda environment to install SLiM into. If you do not use the default name, then it may take longer for slimr to find SLiM (which may increase loading times for the library).

Examples

```
if(interactive()) {
  slim_setup()
}
```

slim_template_info *Get information on templating in a slimr_script*

Description

Returns information on templated variables and their default values in a `slimr_script`

Usage

```
slim_template_info(script_temp)
```

Arguments

`script_temp`	A templated `slimr_script` to retrieve information from

Value

A list of lists. The top-level is named for the blocks in which templated variables exist. For each block with templated variables the element is a list named with all templated variables in that block, and its values are equal to the default values for those variables.

Examples

```
test_sim <- slim_script(
  slim_block_init_minimal(mutation_rate = r_template("mu", 1e-7)),
  slim_block_add_subpops(1, 100),
  slim_block(1, 20, late(), {
    r_output(sim.outputFull(), "out", do_every = 10)
  })
)
```

slim_unload_globals *Unload SLiM Globals from R Global environment*

Description

This will remove any objects added to the global environment by `slim_load_globals`

Usage

```
slim_unload_globals()
```

Value

None

Examples

```
slim_load_globals(c(p = 1))
slim_unload_globals()
```

SM *SpatialMap*

Description

Documentation for SpatialMap class from SLiM

Details

This class represents a ”spatial map”: a grid of values overlaid across a simulated spatial landscape, representing some variable that varies across space. This variable could be something environmental (elevation, temperature), something that affects local dynamics (local carryingcapacity density, local mean dispersal distance), or anything else needed for the model (years since the last wildfire in that location, for example). A new spatial map is created with the defineSpatialMap() method of Subpopulation, because spatial maps are tightly associated with subpopulations; in particular, they must share the spatial bounds of any subpopulation to which they have been added, because the spatial map's grid of values is overlaid onto those spatial bounds. There is a constructor for SpatialMap, but it is only used to copy an existing spatial map: (object<SpatialMap>$)SpatialMap(string$ name, object<SpatialMap>$ map) Creates a new SpatialMap object that is a copy of map, named name. That can be useful if you wish to derive a new spatial map from an existing map, but it is not very commonly used. Usually a new map is created with defineSpatialMap(), often using spatial data from a PNG image file that has been loaded using the Eidos class Image. If you wish to add a spatial map to additional subpopulations, beyond the subpopulation for which the map was originally defined, the Subpopulation method addSpatialMap() can be used to do so, thereby sharing the map across more than one subpopulation. The SpatialMap class was added in SLiM 4.1; before that, spatial maps were a sub-component of Subpopulation. Splitting spatial maps out into their own class provides more flexibility to extend their capabilities in the future. This class has the following methods (functions):

- add
- blend
- changeColors
- changeValues
- divide
- exp
- gridValues
- interpolate
- mapColor

- mapImage
- mapValue
- multiply
- power
- range
- rescale
- sampleImprovedNearbyPoint
- sampleNearbyPoint
- smooth
- subtract

This class has the following properties:

gridDimensions A property of type integer or logical or string or float or string or integer. This property is a constant, so it is not modifiable. **Property Description:** The dimensions of the spatial map's grid of values, in the order of the components of the map's spatiality. For example, a map with spatiality "xz" and a grid of values that is 500 in the "x" dimension by 300 in the "z" dimension would return c(500, 300) for this property.

interpolate A property of type integer or logical or string or float or string or integer. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** Whether interpolation between grid values is enabled (T) or disabled (F). The initial value of this property is set by defineSpatialMap(), but it can be changed. The interpolation performed is linear; for cubic interpolation, use the interpolate() method.

name A property of type integer or logical or string or float or string or integer. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The name of the spatial map, usually as provided to defineSpatialMap(). The names of spatial maps must be unique within any given subpopulation, but the same name may be reused for different spatial maps in different subpopulations. The name is used to identify a map for methods such as spatialMapValue(), and is also used for display in SLiMgui.

spatialBounds A property of type integer or logical or string or float or string or integer. This property is a constant, so it is not modifiable. **Property Description:** The spatial bounds to which the spatial map is aligned. These bounds come from the subpopulation that originally created the map, with the defineSpatialMap() method, and cannot be subsequently changed. All subpopulations that use a given spatial map must match that map's spatial bounds, so that the map does not stretch or shrink relative to its initial configuration. The components of the spatial bounds of a map correspond to the components of the map's spatiality; for example, a map with spatiality "xz" will have bounds (x0, z0, x1, z1); bounds for "y" are not included, since that dimension is not used by the spatial map.

spatiality A property of type integer or logical or string or float or string or integer. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The spatiality of the map: the subset of the model's dimensions that are

used by the spatial map. The spatiality of a map is configured by defineSpatialMap() and cannot subsequently be changed. For example, a 3D model (with dimensionality "xyz") might define a 2D spatial map with spatiality "xz", providing spatial values that do not depend upon the "y" dimension. Often, however, the spatiality of a map will match the dimensionality of the model.

tag A property of type integer or logical or string or float or string or integer. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** A user-defined integer value. The value of tag is initially undefined, and it is an error to try to read it; if you wish it to have a defined value, you must arrange that yourself by explicitly setting its value prior to using it elsewhere in your code. The value of tag is not used by SLiM; it is free for you to use. See also the getValue() and setValue() methods (provided by the Dictionary class; see the Eidos manual), for another way of attaching state to spatial maps.

See Also

Other SpatialMap: `add()`, `blend()`, `changeColors()`, `changeValues()`, `divide()`, `exp()`, `gridValues()`, `interpolate()`, `mapColor()`, `mapImage()`, `mapValue()`, `multiply()`, `power()`, `range()`, `rescale()`, `sampleImprovedNearbyPoint()`, `sampleNearbyPoint()`, `smooth()`, `subtract()`

`smooth` *SLiM method smooth*

Description

Documentation for SLiM function `smooth`, which is a method of the SLiM class `SpatialMap`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
smooth(maxDistance, functionType, ...)
```

Arguments

`maxDistance`	An object of type float. Must be of length 1 (a singleton). See details for description.
`functionType`	An object of type string. Must be of length 1 (a singleton). See details for description.
`...`	An object of type NA. NA See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 717.

Smooths (or blurs, one could say) the values of the spatial map by convolution with a kernel. The kernel is specified with a maximum distance maxDistance (beyond which the kernel cuts off to a value of zero), a kernel type functionType that should be "f", "l", "e", "n", "c", or "t", and additional parameters in the ellipsis ... that depend upon the kernel type and further specify its shape. The target spatial map is returned, to allow easy chaining of operations. The kernel specification is similar to that for the setInteractionType() method of InteractionType, but omits the maximum value of the kernel. Specifically, functionType may be "f", in which case no ellipsis arguments should be supplied; "l", similarly with no ellipsis arguments; "e", in which case the ellipsis should supply a numeric$ lambda (rate) parameter for a negative exponential function; "n", in which case the ellipsis should supply a numeric$ sigma (standard deviation) parameter for a Gaussian function; "c", in which case the ellipsis should supply a numeric$ scale parameter for a Cauchy distribution function; or "t", in which case the ellipsis should supply a numeric$ degrees of freedom and a numeric$ scale parameter for a t-distribution function. See the InteractionType class documentation for discussions of these kernel types. Distance metrics specified to this method, such as maxDistance and the additional kernel shape parameters, are measured in the distance scale of the spatial map - the same distance scale in which the spatial bounds of the map are specified. The operation is performed upon the grid values of the spatial map; distances are internally translated into the scale of the value grid. For non-periodic boundaries, clipping at the edge of the spatial map is done; in a 2D map with no periodic boundaries, for example, the weights of edge and corner grid values are adjusted for their partial (one-half and one-quarter) coverage. For periodic boundaries, the smoothing operation will automatically wrap around based upon the assumption that the grid values at the two connected edges of the periodic boundary have identical values (which they should, since by definition they represent the same position in space). The density scale of the kernel has no effect and will be normalized; this is the reason that smooth(), unlike InteractionType, does not require specification of the maximum value of the kernel. This normalization prevents the kernel from increasing or decreasing the average spatial map value (apart from possible edge effects).

Value

An object of type SpatialMap object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other SpatialMap: `SM`, `add()`, `blend()`, `changeColors()`, `changeValues()`, `divide()`, `exp()`, `gridValues()`, `interpolate()`, `mapColor()`, `mapImage()`, `mapValue()`, `multiply()`, `power()`, `range()`, `rescale()`, `sampleImprovedNearbyPoint()`, `sampleNearbyPoint()`, `subtract()`

`Sp` *Species*

Description

Documentation for Species class from SLiM

Details

This class represents a species in a SLiM simulation. In a single-species model, the single Species instance is defined as a global constant named sim; in multispecies models, each Species instance is defined as a global constant with the same name as the species. This class has the following methods (functions):

- `addSubpop`
- `addSubpopSplit`
- `countOfMutationsOfType`
- `individualsWithPedigreeIDs`
- `killIndividuals`
- `mutationCounts`
- `mutationFrequencies`
- `mutationsOfType`
- `outputFixedMutations`
- `outputFull`
- `outputMutations`
- `readFromPopulationFile`
- `recalculateFitness`
- `registerFitnessEffectCallback`
- `registerMateChoiceCallback`
- `registerModifyChildCallback`
- `registerMutationCallback`
- `registerMutationEffectCallback`
- `registerRecombinationCallback`
- `registerReproductionCallback`
- `registerSurvivalCallback`

- simulationFinished
- skipTick
- subsetMutations
- treeSeqCoalesced
- treeSeqOutput
- treeSeqRememberIndividuals
- treeSeqSimplify

This class has the following properties:

avatar A property of type string. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The avatar string used to represent this species in SLiMgui. Outside of SLiMgui, this property still exists, but is not used by SLiM. Avatars are typically one-character strings, often using an emoji that symbolizes the species. This property is read-only; its value should be set with the avatar parameter of initializeSpecies().

chromosome A property of type Chromosome object. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The Chromosome object used by the species.

chromosomeType A property of type string. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The type of chromosome being simulated by this species; this will be one of "A", "X", or "Y".

color A property of type string. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The color used to display information about this species in SLiMgui. Outside of SLiMgui, this property still exists, but is not used by SLiM. Colors may be specified by name, or with hexadecimal RGB values of the form "#RRGGBB" (see the Eidos manual). This property is read-only; its value should be set with the color parameter of initializeSpecies().

cycle A property of type integer. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** The current cycle count for this species. This counter begins at 1, and increments at the end of every tick in which the species is active. In models with non-overlapping generations, particularly WF models, this can be thought of as a generation counter.

description A property of type string. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** A human-readable string description for the species. By default, this is the empty string, ""; however, it may be set to whatever you wish.

dimensionality A property of type string. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The spatial dimensionality of the simulation for this species, as specified in initializeSLiMOptions(). This will be "" (the empty string) for non-spatial simulations (the default), or "x", "xy", or "xyz", for simulations using those spatial dimensions respectively.

genomicElementTypes A property of type GenomicElementType object. This property is a constant, so it is not modifiable. **Property Description:** The GenomicElementType objects being used in the species.

id A property of type integer. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The identifier for this species. Species identifiers are determined by their declaration order in the script; the first declared species is given an id of 0, the second is given an id of 1, and so forth.

mutationTypes A property of type MutationType object. This property is a constant, so it is not modifiable. **Property Description:** The MutationType objects being used in the species.

mutations A property of type Mutation object. This property is a constant, so it is not modifiable. **Property Description:** The Mutation objects that are currently active in the species.

name A property of type string. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** A human-readable string name for the subpopulation. This is always the declared name of the species, as given in the explicit species declaration in script, and cannot be changed. The name of a species may appear as a label in SLiMgui, and it can be useful in generating output, debugging, and other purposes. See also the description property, which can be changed by the user and used for any purpose.

nucleotideBased A property of type logical. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** If T, the model for this species is nucleotide-based; if F, it is not. See the discussion of the nucleotideBased parameter to initializeSLiMOptions() for discussion.

periodicity A property of type string. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** The spatial periodicity of the simulation for this species, as specified in initializeSLiMOptions(). This will be "" (the empty string) for non-spatial simulations and simulations with no periodic spatial dimensions (the default). Otherwise, it will be a string representing the subset of spatial dimensions that have been declared to be periodic, as specified to initializeSLiMOptions().

scriptBlocks A property of type SLiMEidosBlock object. This property is a constant, so it is not modifiable. **Property Description:** All registered SLiMEidosBlock objects in the simulation that have been declared with this species as their species specifier (not ticks specifier). These will always be callback blocks; callbacks are species-specific, while other types of blocks are not.

sexEnabled A property of type logical. It is of length one (a singleton). This property is a constant, so it is not modifiable. **Property Description:** If T, sex is enabled for this species; if F, individuals are hermaphroditic.

subpopulations A property of type Subpopulation object. This property is a constant, so it is not modifiable. **Property Description:** The Subpopulation instances currently defined in the species.

substitutions A property of type Substitution object. This property is a constant, so it is not modifiable. **Property Description:** A vector of Substitution objects, representing all mutations that have been fixed in this species.

tag A property of type integer. It is of length one (a singleton). This property is a variable, so it is modifiable. **Property Description:** A user-defined integer value. The value of tag is initially undefined, and it is an error to try to read it; if you wish it to have a defined value, you must arrange that yourself by explicitly setting its value prior to

using it elsewhere in your code. The value of tag is not used by SLiM; it is free for you to use. See also the getValue() and setValue() methods (provided by the Dictionary class; see the Eidos manual), for another way of attaching state to the simulation.

See Also

Other Species: `addSubpopSplit()`, `addSubpop()`, `countOfMutationsOfType()`, `individualsWithPedigreeIDs()`, `killIndividuals()`, `mutationCounts()`, `mutationFrequencies()`, `mutationsOfType()`, `outputFixedMutations()`, `outputFull()`, `outputMutations()`, `readFromPopulationFile()`, `recalculateFitness()`, `registerFitnessEffectCallback()`, `registerMateChoiceCallback()`, `registerModifyChildCallback()`, `registerMutationCallback()`, `registerMutationEffectCallback()`, `registerRecombinationCallback()`, `registerReproductionCallback()`, `registerSurvivalCallback()`, `simulationFinished()`, `skipTick()`, `subsetMutations()`, `treeSeqCoalesced()`, `treeSeqOutput()`, `treeSeqRememberIndividuals()`, `treeSeqSimplify()`

`spatialMapColor` *SLiM method spatialMapColor*

Description

Documentation for SLiM function `spatialMapColor`, which is a method of the SLiM class `Subpopulation`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
spatialMapColor(name, value)
```

Arguments

`name`	An object of type string or numeric. Must be of length 1 (a singleton). See details for description.
`value`	An object of type string or numeric. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 745.

This method has been deprecated, and may be removed in a future release of SLiM. In SLiM 4.1 and later, use the SpatialMap method mapColor() instead, and see that method's documentation. (This method differs only in taking a name parameter, which is used to look up the spatial map from those that have been added to the subpopulation.)

Value

An object of type string.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Subpopulation: P, addCloned(), addCrossed(), addEmpty(), addRecombinant(), addSelfed(), addSpatialMap(), cachedFitness(), configureDisplay(), defineSpatialMap(), outputMSSample(), outputSample(), outputVCFSample(), pointDeviated(), pointInBounds(), pointPeriodic(), pointReflected(), pointStopped(), pointUniform(), removeSpatialMap(), removeSubpopulation(), sampleIndividuals(), setCloningRate(), setMigrationRates(), setSelfingRate(), setSexRatio(), setSpatialBounds(), setSubpopulationSize(), spatialMapImage(), spatialMapValue(), subsetIndividuals(), takeMigrants()

---

spatialMapImage *SLiM method spatialMapImage*

Description

Documentation for SLiM function `spatialMapImage`, which is a method of the SLiM class `Subpopulation`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
spatialMapImage(name, width, height, centers, color)
```

Arguments

`name`	An object of type string. Must be of length 1 (a singleton). See details for description.
`width`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`height`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`centers`	An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.
`color`	An object of type logical. Must be of length 1 (a singleton). The default value is `T`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 745.

This method has been deprecated, and may be removed in a future release of SLiM. In SLiM 4.1 and later, use the SpatialMap method mapImage() instead, and see that method's documentation. (This method differs only in taking a name parameter, which is used to look up the spatial map from those that have been added to the subpopulation.)

Value

An object of type Image object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Subpopulation: P, addCloned(), addCrossed(), addEmpty(), addRecombinant(), addSelfed(), addSpatialMap(), cachedFitness(), configureDisplay(), defineSpatialMap(), outputMSSample(), outputSample(), outputVCFSample(), pointDeviated(), pointInBounds(), pointPeriodic(), pointReflected(), pointStopped(), pointUniform(), removeSpatialMap(), removeSubpopulation(), sampleIndividuals(), setCloningRate(), setMigrationRates(), setSelfingRate(), setSexRatio(), setSpatialBounds(), setSubpopulationSize(), spatialMapColor(), spatialMapValue(), subsetIndividuals(), takeMigrants()

---

spatialMapValue *SLiM method spatialMapValue*

Description

Documentation for SLiM function spatialMapValue, which is a method of the SLiM class Subpopulation. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
spatialMapValue(map, point)
```

Arguments

`map`	An object of type string or SpatialMap object. Must be of length 1 (a singleton). See details for description.
`point`	An object of type float. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 745.

Looks up the spatial map specified by map, and uses its mapping machinery (as defined by the gridSize, values, and interpolate parameters to defineSpatialMap()) to translate the coordinates of point into a corresponding map value. The parameter map may specify the map either as a SpatialMap object, or by its string name; in either case, the map must have been added to the subpopulation. The length of point must be equal to the spatiality of the spatial map; in other words, for a spatial map with spatiality "xz", point must be of length 2, specifying the x and z coordinates of the point to be evaluated. Interpolation will automatically be used if it was enabled for the spatial map. Point coordinates are clamped into the range defined by the spatial boundaries, even if the spatial boundaries are periodic; use pointPeriodic() to wrap the point coordinates first if desired. See the documentation for defineSpatialMap() for information regarding the details of value mapping. Beginning in SLiM 3.3, point may contain more than one point to be looked up. In this case, the length of point must be an exact multiple of the spatiality of the spatial map; for a spatial map with spatiality "xz", for example, the length of point must be an exact multiple of 2, and successive pairs of elements from point (elements 0 and 1, then elements 2 and 3, etc.) will be taken as the x and z coordinates of the points to be evaluated. This allows spatialMapValue() to be used in a vectorized fashion. The mapValue() method of SpatialMap provides the same functionality directly on the SpatialMap class; spatialMapValue() is provided on Subpopulation partly for backward compatibility, but also for convenience for the most common usage case of spatial maps.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Subpopulation: P, addCloned(), addCrossed(), addEmpty(), addRecombinant(), addSelfed(), addSpatialMap(), cachedFitness(), configureDisplay(), defineSpatialMap(),

outputMSSample(), outputSample(), outputVCFSample(), pointDeviated(), pointInBounds(), pointPeriodic(), pointReflected(), pointStopped(), pointUniform(), removeSpatialMap(), removeSubpopulation(), sampleIndividuals(), setCloningRate(), setMigrationRates(), setSelfingRate(), setSexRatio(), setSpatialBounds(), setSubpopulationSize(), spatialMapColor(), spatialMapImage(), subsetIndividuals(), takeMigrants()

`speciesWithIDs` *SLiM method speciesWithIDs*

Description

Documentation for SLiM function `speciesWithIDs`, which is a method of the SLiM class `Community`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
speciesWithIDs(ids)
```

Arguments

`ids`	An object of type integer. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 668.

Find and return the Species objects with id values matching the values in ids. If no matching Species object can be found with a given id, an error results.

Value

An object of type Species object.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Community: Co, createLogFile(), deregisterScriptBlock(), genomicElementTypesWithIDs(), interactionTypesWithIDs(), mutationTypesWithIDs(), outputUsage(), registerEarlyEvent(), registerFirstEvent(), registerInteractionCallback(), registerLateEvent(), rescheduleScriptBlock(), scriptBlocksWithIDs(), simulationFinished(), subpopulationsWithIDs(), usage()

`SS` *SLiMSim was a class used prior to v4.0 in SLiM. It remains in 'slimr' for backwards compatibility but its use is now deprecated.*

Description

SLiMSim was a class used prior to v4.0 in SLiM. It remains in 'slimr' for backwards compatibility but its use is now deprecated.

`strength` *SLiM method strength*

Description

Documentation for SLiM function `strength`, which is a method of the SLiM class `InteractionType`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
strength(receiver, exerters)
```

Arguments

`receiver`	An object of type Individual object. Must be of length 1 (a singleton). See details for description.
`exerters`	An object of type null or Individual object. The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 698.

Returns a vector containing the interaction strengths exerted upon receiver by the individuals in exerters. If exerters is NULL (the default), then a vector of the interaction strengths exerted by all individuals in the subpopulation of receiver (including receiver itself, with a strength of 0.0) is returned; this case may be handled much more efficiently than if a vector of all individuals in the subpopulation is explicitly provided. Otherwise, all individuals in exerters must belong to a single subpopulation (but not necessarily the same subpopulation as receiver). The evaluate() method must have been previously called for the receiver and exerter subpopulations, and positions saved at evaluation time will be used. If the strengths of interactions exerted by a single individual upon multiple individuals are needed instead (the inverse of what this method provides), multiple calls to this method will be necessary, one per pairwise interaction queried; the interaction engine is not optimized for the inverse case, and so it will likely be quite slow to compute. If the interaction is reciprocal and has the same receiver and exerter constraints, the opposite query should provide identical results in a single efficient call (because then the interactions exerted are equal to the interactions received); otherwise, the best approach might be to define a second interaction type representing the inverse interaction that you wish to be able to query efficiently.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other InteractionType: IT, clippedIntegral(), distanceFromPoint(), distance(), drawByStrength(), evaluate(), interactingNeighborCount(), interactionDistance(), localPopulationDensity(), nearestInteractingNeighbors(), nearestNeighborsOfPoint(), nearestNeighbors(), neighborCountOfPoint(), neighborCount(), setConstraints(), setInteractionFunction(), testConstraints(), totalOfNeighborStrengths(), unevaluate()

`subpopulationsWithIDs`
SLiM method subpopulationsWithIDs

Description

Documentation for SLiM function `subpopulationsWithIDs`, which is a method of the SLiM class `Community`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
subpopulationsWithIDs(ids)
```

Arguments

`ids`	An object of type integer. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 668.

Find and return the Subpopulation objects with id values matching the values in ids. If no matching Subpopulation object can be found with a given id, an error results.

Value

An object of type Subpopulation object.

Copyright

Author(s)

Benjamin C Haller (`<bhaller@benhaller.com>`) and Philipp W Messer (`<messer@cornell.edu>`)

See Also

Other Community: `Co`, `createLogFile()`, `deregisterScriptBlock()`, `genomicElementTypesWithIDs()`, `interactionTypesWithIDs()`, `mutationTypesWithIDs()`, `outputUsage()`, `registerEarlyEvent()`, `registerFirstEvent()`, `registerInteractionCallback()`, `registerLateEvent()`, `rescheduleScriptBlock()`, `scriptBlocksWithIDs()`, `simulationFinished()`, `speciesWithIDs()`, `usage()`

`subsetIndividuals` *SLiM method subsetIndividuals*

Description

Documentation for SLiM function `subsetIndividuals`, which is a method of the SLiM class `Subpopulation`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
subsetIndividuals(
  exclude,
  sex,
  tag,
  minAge,
  maxAge,
  migrant,
  tagL0,
  tagL1,
  tagL2,
  tagL3,
  tagL4
)
```

Arguments

`exclude`	An object of type null or Individual object. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`sex`	An object of type null or string. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`tag`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`minAge`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`maxAge`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`migrant`	An object of type null or logical. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`tagL0`	An object of type null or logical. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`tagL1`	An object of type null or logical. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

`tagL2`	An object of type null or logical. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`tagL3`	An object of type null or logical. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`tagL4`	An object of type null or logical. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 746.

Returns a vector of individuals subset from the individuals in the target subpopulation. The parameters specify constraints upon the subset of individuals that will be returned. Parameter exclude, if non- NULL, may specify a specific individual that should not be included (typically the focal individual in some operation). Parameter sex, if non-NULL, may specify a sex ("M" or "F") for the individuals to be returned, in sexual models. Parameter tag, if non-NULL, may specify a tag property value for the individuals to be returned. Parameters minAge and maxAge, if non-NULL, may specify a minimum or maximum age for the individuals to be returned, in nonWF models. Parameter migrant, if non-NULL, may specify a required value for the migrant property of the individuals to be returned (so T will require that individuals be migrants, F will require that they not be). Finally, parameters tagL0, tagL1, tagL2, tagL3, and tagL4, if non-NULL, may specify a required value (T or F) for the corresponding properties (tagL0, tagL1, tagL2, tagL3, and tagL4) of the individuals to be returned. Note that if any tag/tagL parameter is specified as non-NULL, that tag/tagL property must have a defined value for every individual in the subpopulation, otherwise an error may result (although this requirement will not necessarily be checked comprehensively by this method in every invocation). This method is shorthand for getting the individuals property of the subpopulation, and then using operator [] to select only individuals with the desired properties; besides being much simpler than the equivalent Eidos code, it is also much faster. See sampleIndividuals() for a similar method that returns a sample taken from a chosen subset of individuals.

Value

An object of type Individual object.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Subpopulation: P, addCloned(), addCrossed(), addEmpty(), addRecombinant(), addSelfed(), addSpatialMap(), cachedFitness(), configureDisplay(), defineSpatialMap(), outputMSSample(), outputSample(), outputVCFSample(), pointDeviated(), pointInBounds(), pointPeriodic(), pointReflected(), pointStopped(), pointUniform(), removeSpatialMap(), removeSubpopulation(), sampleIndividuals(), setCloningRate(), setMigrationRates(), setSelfingRate(), setSexRatio(), setSpatialBounds(), setSubpopulationSize(), spatialMapColor(), spatialMapImage(), spatialMapValue(), takeMigrants()

`subsetMutations` *SLiM method subsetMutations*

Description

Documentation for SLiM function `subsetMutations`, which is a method of the SLiM class Species. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
subsetMutations(exclude, mutType, position, nucleotide, tag, id)
```

Arguments

`exclude`	An object of type null or Mutation object. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`mutType`	An object of type null or integer or MutationType object. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`position`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`nucleotide`	An object of type null or integer or string. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`tag`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.
`id`	An object of type null or integer. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 727.

Returns a vector of mutations subset from the list of all active mutations in the species (as would be provided by the mutations property). The parameters specify constraints

upon the subset of mutations that will be returned. Parameter exclude, if non-NULL, may specify a specific mutation that should not be included (typically the focal mutation in some operation). Parameter mutType, if non- NULL, may specify a mutation type for the mutations to be returned (as either a MutationType object or an integer identifier). Parameter position, if non-NULL, may specify a base position for the mutations to be returned. Parameter nucleotide, if non-NULL, may specify a nucleotide for the mutations to be returned (either as a string, "A" / "C" / "G" / "T", or as an integer, 0 / 1 / 2 / 3 respectively). Parameter tag, if non-NULL, may specify a tag value for the mutations to be returned. Parameter id, if non-NULL, may specify a required value for the id property of the mutations to be returned. This method is shorthand for getting the mutations property of the subpopulation, and then using operator [] to select only mutations with the desired properties; besides being much simpler than the equivalent Eidos code, it is also much faster. Note that if you only need to select on mutation type, the mutationsOfType() method will be even faster.

Value

An object of type Mutation object.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Species: Sp, addSubpopSplit(), addSubpop(), countOfMutationsOfType(), individualsWithPedigreeID
killIndividuals(), mutationCounts(), mutationFrequencies(), mutationsOfType(),
outputFixedMutations(), outputFull(), outputMutations(), readFromPopulationFile(),
recalculateFitness(), registerFitnessEffectCallback(), registerMateChoiceCallback(),
registerModifyChildCallback(), registerMutationCallback(), registerMutationEffectCallback(),
registerRecombinationCallback(), registerReproductionCallback(), registerSurvivalCallback(),
simulationFinished(), skipTick(), treeSeqCoalesced(), treeSeqOutput(), treeSeqRememberIndividuals()
treeSeqSimplify()

`subtract` *SLiM method subtract*

Description

Documentation for SLiM function `subtract`, which is a method of the SLiM class `SpatialMap`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
subtract(x)
```

Arguments

`x`	An object of type integer or float or SpatialMap object. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 717.

Subtracts x from the spatial map. One possibility is that x is a singleton integer or float value; in this case, x is subtracted from each grid value of the target spatial map. Another possibility is that x is an integer or float vector/matrix/array of the same dimensions as the target spatial map's grid; in this case, each value of x is subtracted from the corresponding grid value of the target spatial map. The third possibility is that x is itself a (singleton) spatial map; in this case, each grid value of x is subtracted from the corresponding grid value of the target spatial map (and thus the two spatial maps must match in their spatiality, their spatial bounds, and their grid dimensions). The target spatial map is returned, to allow easy chaining of operations.

Value

An object of type SpatialMap object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (`<bhaller@benhaller.com>`) and Philipp W Messer (`<messer@cornell.edu>`)

See Also

Other SpatialMap: `SM`, `add()`, `blend()`, `changeColors()`, `changeValues()`, `divide()`, `exp()`, `gridValues()`, `interpolate()`, `mapColor()`, `mapImage()`, `mapValue()`, `multiply()`,

power(), range(), rescale(), sampleImprovedNearbyPoint(), sampleNearbyPoint(), smooth()

`summarizeIndividuals` *SLiM method summarizeIndividuals*

Description

Documentation for SLiM function `summarizeIndividuals`, which is a method of the SLiM class `SLiMBuiltin`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
summarizeIndividuals(
  individuals,
  dim,
  spatialBounds,
  operation,
  empty,
  perUnitArea,
  spatiality
)
```

Arguments

`individuals`	An object of type Individual object. See details for description.
`dim`	An object of type integer. See details for description.
`spatialBounds`	An object of type numeric. See details for description.
`operation`	An object of type string. Must be of length 1 (a singleton). See details for description.
`empty`	An object of type null or logical or integer or float. Must be of length 1 (a singleton). The default value is `0.0`. See details for description.
`perUnitArea`	An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.
`spatiality`	An object of type null or string. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 753.

Returns a vector, matrix, or array that summarizes spatial patterns of information related to the individuals in individuals. In essence, those individuals are assigned into bins according to their spatial position, and then a summary value for each bin is calculated based upon the

individuals each bin contains. The individuals might be binned in one dimension (resulting in a vector of summary values), in two dimensions (resulting in a matrix), or in three dimensions (resulting in an array). Typically the spatiality of the result (the dimensions into which the individuals are binned) will match the dimensionality of the model, as indicated by the default value of NULL for the optional spatiality parameter; for example, a two-dimensional ("xy") model would by default produce a two-dimensional matrix as a summary. However, a spatiality that is more restrictive than the model dimensionality may be passed; for example, in a two-dimensional ("xy") model a spatiality of "y" could be passed to summarize individuals into a vector, rather than a matrix, assigning them to bins based only upon their y position (i.e., the value of their y property). Whatever spatiality is chosen, the parameter dim provides the dimensions of the desired result, in the same form that the dim() function does: first the number of rows, then the number of columns, and then the number of planes, as needed (see the Eidos manual for discussion of matrices, arrays, and dim()). The length of dims must match the requested spatiality; for spatiality "xy", for example, dims might be c(50,100) to request that the returned matrix have 50 rows and 100 columns. The result vector/matrix/array is in the correct orientation to be directly usable as a spatial map, by passing it to the defineSpatialMap() method of Subpopulation. For further discussion of dimensionality and spatiality, see section 25.1 on initializeInteractionType(), and section 25.8 on InteractionType. The spatialBounds parameter defines the spatial boundaries within which the individuals are binned. Typically this is the spatial bounds of a particular subpopulation, within which the individuals reside; for individuals in p1, for example, you would likely pass p1.spatialBounds for this. However, this is not required; individuals may come from any or all subpopulations in the model, and spatialBounds may be any bounds of non-zero area (if an individual falls outside of the given spatial bounds, it is excluded, as if it were not in individuals at all). If you have multiple subpopulations that conceptually reside within the same overall coordinate space, for example, that can be accommodated here. The bounds are supplied in the dimensionality of the model, in the same form as for Subpopulation; for an "xy" model, for example, they are supplied as a four-element vector of the form c(x0, y0, x1, y1) even if the summary is being produced with spatiality "y". To produce the result, a grid with dimensions defined by dims is conceptually stretched out across the given spatial bounds, such that the centers of the edge and corner grid squares are aligned with the limits of the spatial bounds. This matches the way that defineSpatialMap() defines its maps; see section 16.11 for illustration. The particular summary produced depends upon the parameters operation and empty. Consider a single grid square represented by a single element in the result. That grid square contains zero or more of the individuals in individuals. If it contains zero individuals and empty is not NULL, the empty value is used for the result, regardless of operation, providing specific, separate control over the treatment of empty grid squares. If empty is NULL, this separate control over the treatment of empty grid squares is declined; empty grid squares will be handled through the standard mechanism described next. In all other cases for the given grid square - when it contains more than zero individuals, or when empty is NULL - operation is executed as an Eidos lambda, a small snippet of code, supplied as a singleton string, that is executed in a manner similar to a function call. Within the execution of the operation lambda, a constant named individuals is defined to be the focal individuals being evaluated - all of the individuals within that grid square. This lambda should evaluate to a singleton logical, integer, or float value, comprising the result value for the grid square; these types will all be coerced to float (T being 1 and F being 0). Two examples may illustrate the use of empty and operation. To produce a summary indicating

presence/absence, simply use the default of 0.0 for empty, and ”1.0; ” (or ”1;”, or ”T;”) for operation. This will produce 0.0 for empty grid squares, and 1.0 for those that contain at least one individual. Note that the use of empty is essential here, because operation doesn't even check whether individuals are present or not. To produce a summary with a count of the number of individuals in each grid square, again use the default of 0.0 for empty, but now use an operation of ”individuals.size();”, counting the number of individuals in each grid square. In this case, empty could be NULL instead and operation would still produce the correct result; but using empty makes summarizeIndividuals() more efficient since it allows the execution of operation to be skipped for those squares. Lambdas are not limited in their complexity; they can use if, for, etc., and can call methods and functions. A typical operation to compute the mean phenotype in a quantitative genetic model that stores phenotype values in tagF, for example, would be ”mean(individuals.tagF);”, and this is still quite simple compared to what is possible. However, keep in mind that the lambda will be evaluated for every grid cell (or at least those that are non-empty), so efficiency can be a concern, and you may wish to pre-calculate values shared by all of the lambda calls, making them available to your lambda code using defineGlobal() or defineConstant(). There is one last twist, if perUnitArea is T: values are divided by the area (or length, in 1D, or volume, in 3D) that their corresponding grid cell comprises, so that each value is in units of ”per unit area” (or ”per unit length”, or ”per unit volume”). The total area of the grid is defined by the spatial bounds, and the area of a given grid cell is defined by the portion of the spatial bounds that is within that cell. This is not the same for all grid cells; grid cells that fall partially outside spatialBounds (because, remember, the centers of the edge/corner grid cells are aligned with the limits of spatialBounds) will have a smaller area inside the bounds. For an ”xy” spatiality summary, for example, corner cells have only a quarter of their area inside spatialBounds, while edge elements have half of their area inside spatialBounds; for purposes of perUnitArea, then, their respective areas are ¼ and ½ the area of an interior grid cell. By default, perUnitArea is F, and no scaling is performed. Whether you want perUnitArea to be F or T depends upon whether the summary you are producing is, conceptually, ”per unit area”, such as density (individuals per unit area) or local competition strength (total interaction strength per unit area), or is not, such as ”mean individual age”, or ”maximum tag value”. For the previous example of counting individuals with an operation of ”individuals.size();”, a value of F for perUnitArea (the default) will produce a simple count of individuals in each grid square, whereas with T it would produce the density of individuals in each grid square.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other SLiMBuiltin: SB, calcFST(), calcHeterozygosity(), calcInbreedingLoad(), calcPairHeterozygosity(
calcVA(), calcWattersonsTheta(), codonsToAminoAcids(), mm16To256(), mmJukesCantor(),
mmKimura(), nucleotideCounts(), nucleotideFrequencies(), nucleotidesToCodons(),
treeSeqMetadata()

Examples

```
## This just brings up the documentation:
summarizeIndividuals()
```

---

sumOfMutationsOfType *SLiM method sumOfMutationsOfType*

Description

Documentation for SLiM function sumOfMutationsOfType, which is a method of the SLiM class Genome. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Documentation for SLiM function sumOfMutationsOfType, which is a method of the SLiM class Individual. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
sumOfMutationsOfType(mutType)

sumOfMutationsOfType(mutType)
```

Arguments

mutType	An object of type integer or MutationType object. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 676.

Returns the sum of the selection coefficients of all mutations that are of the type specified by mutType, out of all of the mutations in the genome. This is often useful in models that use a particular mutation type to represent QTLs with additive effects; in that context, sumOfMutationsOfType() will provide the sum of the additive effects of the QTLs for the given mutation type. This method is provided for speed; it is much faster than the corresponding Eidos code. Note that this method also exists on Individual, for cases in which the sum across both genomes of an individual is desired.

Documentation for this function can be found in the official SLiM manual: page 685.

Returns the sum of the selection coefficients of all mutations that are of the type specified by mutType, out of all of the mutations in the genomes of the individual. This is often useful in models that use a particular mutation type to represent QTLs with additive effects; in that context, sumOfMutationsOfType() will provide the sum of the additive effects of the QTLs for the given mutation type. This method is provided for speed; it is much faster than the corresponding Eidos code. Note that this method also exists on Genome, for cases in which the sum for just one genome is desired.

Value

An object of type float. Return will be of length 1 (a singleton)

An object of type float. Return will be of length 1 (a singleton)

Copyright

This is documentation for a function in the SLiM software, and has been reproduced from the official manual, which can be found here: http://benhaller.com/slim/SLiM_Manual.pdf. This documentation is Copyright © 2016-2020 Philipp Messer. All rights reserved. More information about SLiM can be found on the official website: https://messerlab.org/slim/

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Genome: G, addMutations(), addNewDrawnMutation(), addNewMutation(), containsMarkerMutation(), containsMutations(), countOfMutationsOfType(), mutationCountsInGenomes(), mutationFrequenciesInGe mutationsOfType(), nucleotides(), outputMS(), outputVCF(), output(), positionsOfMutationsOfType(), readFromMS(), readFromVCF(), removeMutations()

Other Individual: In, containsMutations(), countOfMutationsOfType(), relatedness(), setSpatialPosition(), sharedParentCount(), uniqueMutationsOfType()

`survival` *SLiM survival() callback*

Description

The survival() callback will be called during the selection phase of the tick cycle of nonWF models, during the tick(s) in which it is active. By default it will be called once per individual in the entire population (whether slated for survival or not); it may optionally be restricted to apply only to individuals in a specified subpopulation, using the <subpop-id> specifier. (In multispecies models, the definition must be preceded by a species specification as usual.) When a survival() callback is called, a focal individual has already been evaluated by SLiM regarding its survival; a final fitness value for the individual has been calculated, and a random uniform draw in [0,1] has been generated that determines whether the individual is to survive (a draw less than the individual's fitness) or die (a draw greater than or equal to the individual's fitness). The focal individual is provided to the callback, as is the subpopulation in which it resides. Furthermore, the preliminary decision (whether the focal individual will survive or not), the focal individual's fitness, and the random draw made by SLiM to determine survival are also provided to the callback. The callback may return NULL to accept SLiM's decision, or may return T to indicate that the individual should survive, or F to indicate that it should die, regardless of its fitness and the random deviate drawn. The callback may also return a singleton Subpopulation object to indicate the individual should remain alive but should be moved to that subpopulation (note that calling takeMigrants() during the survival phase is illegal, because SLiM is busy modifying the population's internal state). For more details see SLiM Manual: page 727

Usage

```
survival(subpop_id)
```

Arguments

`subpop_id`	The id(s) of the subpopulation(s) to which this callback should apply. Can be an integer 1, 2, etc., or character "p1", "p2", etc.

Details

Global variables available in reproduction callbacks:

individual The focal parent that is generating a gamete

subpop The subpopulation to which the focal parent belongs

surviving A logical value indicating SLiM's preliminary decision (T == survival)

fitness The focal individual's fitness

draw SLiM's random uniform deviate, which determined the preliminary decision

Value

None

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other callbacks: `early()`, `first()`, `fitnessEffect()`, `fitness()`, `initialize()`, `interaction()`, `late()`, `mateChoice()`, `modifyChild()`, `mutationEffect()`, `mutation()`, `recombination()`, `reproduction()`, `slim_callbacks()`

Examples

```
# From page 461 of the SLiM Manual
slim_block(survival(p1), {
  # move dying males into cold storage in case they have mated
  if (!surviving) {
    if (individual.sex == "M") {
      return(p1000);
    }
    return(NULL);
  }
})
```

`takeMigrants` *SLiM method takeMigrants*

Description

Documentation for SLiM function `takeMigrants`, which is a method of the SLiM class `Subpopulation`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
takeMigrants(migrants)
```

Arguments

`migrants`	An object of type Individual object. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 746.

Immediately moves the individuals in migrants to the target subpopulation (removing them from their previous subpopulation). Individuals in migrants that are already in the target subpopulation are unaffected. Note that the indices and order of individuals and genomes in both the target and source subpopulations will change unpredictably as a side effect of this method. Note that this method is only for use in nonWF models, in which migration is managed manually by the model script. In WF models, migration is managed automatically by the SLiM core based upon the migration rates set for each subpopulation with setMigrationRates().

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Subpopulation: P, addCloned(), addCrossed(), addEmpty(), addRecombinant(), addSelfed(), addSpatialMap(), cachedFitness(), configureDisplay(), defineSpatialMap(), outputMSSample(), outputSample(), outputVCFSample(), pointDeviated(), pointInBounds(), pointPeriodic(), pointReflected(), pointStopped(), pointUniform(), removeSpatialMap(), removeSubpopulation(), sampleIndividuals(), setCloningRate(), setMigrationRates(), setSelfingRate(), setSexRatio(), setSpatialBounds(), setSubpopulationSize(), spatialMapColor(), spatialMapImage(), spatialMapValue(), subsetIndividuals()

---

testConstraints *SLiM method testConstraints*

Description

Documentation for SLiM function testConstraints, which is a method of the SLiM class InteractionType. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
testConstraints(individuals, constraints, returnIndividuals)
```

Arguments

`individuals`	An object of type Individual object. See details for description.
`constraints`	An object of type string. Must be of length 1 (a singleton). See details for description.
`returnIndividuals`	An object of type logical. Must be of length 1 (a singleton). The default value is `F`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 698.

Tests the individuals in the parameter individuals against the interaction constraints specified by constraints. The value of constraints may be ”receiver” to use the receiver constraints, or ”exerter” to use the exerter constraints. If returnIndividuals is F (the default), a logical vector will be returned, with T values indicating that the corresponding individual satisfied the constraints, F values indicating that it did not. If returnIndividuals is T, an object vector of class Individual will be returned containing only those elements of individuals that satisfied the constraints (in the same order as individuals). Note that unlike most queries, the InteractionType does not need to have been evaluated before calling this method, and the individuals passed in need not belong to a single population or even a single species. This method can be useful for narrowing a vector of individuals down to just those that satisfy constraints. Outside the context of InteractionType, similar functionality is provided by the Subpopulation method subsetIndividuals(). Note that the use of testConstraints() is somewhat rare; usually, queries are evaluated across a vector of individuals, each of which might or might not satisfy the defined constraints. Individuals that do not satisfy constraints do not participate in interactions, so their interaction strength with other individuals will simply be zero. See the setConstraints() method to set up constraints, as well as the sexSegregation parameter to initializeInteractionType(). Note that if the constraints tested involve tag values (including tagL0 / tagL1 / tagL2 / tagL3 / tagL4), the corresponding property or properties of the tested individuals must be defined (i.e., must have been set to a value), or an error will result because the constraints cannot be applied.

Value

An object of type logical or Individual object.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other InteractionType: IT, clippedIntegral(), distanceFromPoint(), distance(), drawByStrength(), evaluate(), interactingNeighborCount(), interactionDistance(), localPopulationDensity(), nearestInteractingNeighbors(), nearestNeighborsOfPoint(), nearestNeighbors(), neighborCountOfPoint(), neighborCount(), setConstraints(), setInteractionFunction(), strength(), totalOfNeighborStrengths(), unevaluate()

---

totalOfNeighborStrengths

SLiM method totalOfNeighborStrengths

Description

Documentation for SLiM function totalOfNeighborStrengths, which is a method of the SLiM class InteractionType. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
totalOfNeighborStrengths(receivers, exerterSubpop)
```

Arguments

receivers	An object of type Individual object. See details for description.
exerterSubpop	An object of type null or Subpopulation object. Must be of length 1 (a singleton). The default value is NULL. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 699.

Returns a vector of the total interaction strength felt by each individual in receivers by the exerters in exerterSubpop (or, if that is NULL, then by all individuals in the receiver's subpopulation). The receivers parameter does not need to be a singleton; indeed, it can be a vector of all of the individuals in a given subpopulation. All of the receivers must belong to a single subpopulation, and all of the exerters must belong to a single subpopulation, but those two subpopulations do not need to be the same. The evaluate() method must have been previously called for the receiver and exerter subpopulations, and positions saved at evaluation time will be used. If the InteractionType is nonspatial, this method may not be called. For one individual, this is essentially the same as calling nearestInteractingNeighbors() with a large count so as to obtain the complete vector of all interacting neighbors, calling strength() for each of those interactions to get each interaction strength, and adding

those interaction strengths together with sum(). This method is much faster than that implementation, however, since all of that work is done as a single operation. Also, totalOfNeighborStrengths() can total up interactions for more than one receiver in a single vectorized call. Similarly, for one individual this is essentially the same as calling strength() to get the interaction strengths between a receiver and all individuals in the exerter subpopulation, and then calling sum(). Again, this method should be much faster, since this algorithm looks only at neighbors, whereas calling strength() directly assesses interaction strengths with all other individuals. This will make a particularly large difference when the subpopulation size is large and the maximum distance of the InteractionType is small. See localPopulationDensity() for a related method that calculates the total interaction strength divided by the amount of "interaction field" present for an individual (i.e., the integral of the interaction function clipped to the spatial bounds of the subpopulation) to provide an estimate of the "interaction density" felt by an individual.

Value

An object of type float.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other InteractionType: IT, clippedIntegral(), distanceFromPoint(), distance(), drawByStrength(), evaluate(), interactingNeighborCount(), interactionDistance(), localPopulationDensity(), nearestInteractingNeighbors(), nearestNeighborsOfPoint(), nearestNeighbors(), neighborCountOfPoint(), neighborCount(), setConstraints(), setInteractionFunction(), strength(), testConstraints(), unevaluate()

---

treeSeqCoalesced *SLiM method treeSeqCoalesced*

Description

Documentation for SLiM function treeSeqCoalesced, which is a method of the SLiM class Species. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
treeSeqCoalesced(void)
```

Arguments

`void`	An object of type . See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 728.

Returns the coalescence state for the recorded tree sequence at the last simplification. The returned value is a logical singleton flag, T to indicate that full coalescence was observed at the last treesequence simplification (meaning that there is a single ancestral individual that roots all ancestry trees at all sites along the chromosome - although not necessarily the same ancestor at all sites), or F if full coalescence was not observed. For simple models, reaching coalescence may indicate that the model has reached an equilibrium state, but this may not be true in models that modify the dynamics of the model during execution by changing migration rates, introducing new mutations programmatically, dictating non-random mating, etc., so be careful not to attach more meaning to coalescence than it is due; some models may require burn-in beyond coalescence to reach equilibrium, or may not have an equilibrium state at all. Also note that some actions by a model, such as adding a new subpopulation, may cause the coalescence state to revert from T back to F (at the next simplification), so a return value of T may not necessarily mean that the model is coalesced at the present moment - only that it was coalesced at the last simplification. This method may only be called if tree sequence recording has been turned on with initializeTreeSeq(); in addition, checkCoalescence=T must have been supplied to initializeTreeSeq(), so that the necessary work is done during each tree-sequence simplification. Since this method does not perform coalescence checking itself, but instead simply returns the coalescence state observed at the last simplification, it may be desirable to call treeSeqSimplify() immediately before treeSeqCoalesced() to obtain up-to-date information. However, the speed penalty of doing this in every tick would be large, and most models do not need this level of precision; usually it is sufficient to know that the model has coalesced, without knowing whether that happened in the current tick or in a recent preceding tick.

Value

An object of type logical. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Species: Sp, addSubpopSplit(), addSubpop(), countOfMutationsOfType(), individualsWithPedigreeID
killIndividuals(), mutationCounts(), mutationFrequencies(), mutationsOfType(),
outputFixedMutations(), outputFull(), outputMutations(), readFromPopulationFile(),
recalculateFitness(), registerFitnessEffectCallback(), registerMateChoiceCallback(),
registerModifyChildCallback(), registerMutationCallback(), registerMutationEffectCallback(),
registerRecombinationCallback(), registerReproductionCallback(), registerSurvivalCallback(),
simulationFinished(), skipTick(), subsetMutations(), treeSeqOutput(), treeSeqRememberIndividuals(),
treeSeqSimplify()

`treeSeqMetadata` *SLiM method treeSeqMetadata*

Description

Documentation for SLiM function `treeSeqMetadata`, which is a method of the SLiM class SLiMBuiltin. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
treeSeqMetadata(filePath, userData)
```

Arguments

`filePath`	An object of type string or logical. Must be of length 1 (a singleton). See details for description.
`userData`	An object of type string or logical. Must be of length 1 (a singleton). The default value is `T`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 755.

Returns a Dictionary containing top-level metadata from the .trees (tree-sequence) file at filePath. If userData is T (the default), the top-level metadata under the SLiM/user_metadata key is returned; this is the same metadata that can optionally be supplied to treeSeqOutput() in its metadata parameter, so it makes it easy to recover metadata that you attached to the tree sequence when it was saved. If userData is F, the entire top-level metadata Dictionary object is returned; this can be useful for examining the values of other keys under the SLiM key (see section 27.4), or values inside the top-level dictionary itself that might have been placed there by msprime or other software. This function can be used to read in parameter values or other saved state (tag property values, for example), in order to resuscitate the complete state of a simulation that was written to a .trees file. It could be used for more esoteric purposes too, such as to search through .trees files in a directory (with the help of the Eidos function filesAtPath()) to find those files that satisfy some metadata criterion.

Value

An object of type Dictionary object. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other SLiMBuiltin: SB, calcFST(), calcHeterozygosity(), calcInbreedingLoad(), calcPairHeterozygosity(
calcVA(), calcWattersonsTheta(), codonsToAminoAcids(), mm16To256(), mmJukesCantor(),
mmKimura(), nucleotideCounts(), nucleotideFrequencies(), nucleotidesToCodons(),
summarizeIndividuals()

Examples

```
## This just brings up the documentation:
treeSeqMetadata()
```

---

treeSeqOutput *SLiM method treeSeqOutput*

Description

Documentation for SLiM function `treeSeqOutput`, which is a method of the SLiM class Species. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
treeSeqOutput(path, simplify, includeModel, metadata)
```

Arguments

`path`	An object of type string. Must be of length 1 (a singleton). See details for description.
`simplify`	An object of type logical. Must be of length 1 (a singleton). The default value is `T`. See details for description.
`includeModel`	An object of type logical. Must be of length 1 (a singleton). The default value is `T`. See details for description.
`metadata`	An object of type null. Must be of length 1 (a singleton). The default value is `NULL`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 728.

Outputs the current tree sequence recording tables to the path specified by path. This method may only be called if tree sequence recording has been turned on with initializeTreeSeq(). If simplify is T (the default), simplification will be done immediately prior to output; this is almost always desirable, unless a model wishes to avoid simplification entirely. (Note that if simplification is not done, then all genomes since the last simplification will be marked as samples in the resulting tree sequence.) A binary tree sequence file will be written to the specified path; a filename extension of .trees is suggested for this type of file. Normally, the full SLiM script used to generate the tree sequence is written out to the provenance entry of the tree sequence file, to the model subkey of the parameters top-level key. Supplying F for includeModel suppresses output of the full script; see section 27.4.6 for further discussion. A Dictionary object containing user-generated metadata may be supplied with the metadata parameter. If present, this dictionary will be serialized as JSON and attached to the saved tree sequence under a key named user_metadata, within the SLiM key (see section 27.4.5). If tskit is used to read the tree sequence in Python, this metadata will automatically be deserialized and made available at ts.metadata["SLiM"]["user_metadata"]. This metadata dictionary is not used by SLiM, or by pyslim, tskit, or msprime; you may use it for any purpose you wish. Note that metadata may actually be any subclass of Dictionary, such as a DataFrame. It can even be a Species object such as sim, or a LogFile instance; however, only the keys and values contained by the object's Dictionary superclass state will be serialized into the metadata (properties of the subclass will be ignored). This metadata dictionary can be recovered from the saved file using the treeSeqMetadata() function.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Species: Sp, addSubpopSplit(), addSubpop(), countOfMutationsOfType(), individualsWithPedigreeID
killIndividuals(), mutationCounts(), mutationFrequencies(), mutationsOfType(),
outputFixedMutations(), outputFull(), outputMutations(), readFromPopulationFile(),
recalculateFitness(), registerFitnessEffectCallback(), registerMateChoiceCallback(),
registerModifyChildCallback(), registerMutationCallback(), registerMutationEffectCallback(),
registerRecombinationCallback(), registerReproductionCallback(), registerSurvivalCallback(),
simulationFinished(), skipTick(), subsetMutations(), treeSeqCoalesced(), treeSeqRememberIndividuals
treeSeqSimplify()

---

treeSeqRememberIndividuals

SLiM method treeSeqRememberIndividuals

Description

Documentation for SLiM function `treeSeqRememberIndividuals`, which is a method of the SLiM class `Species`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
treeSeqRememberIndividuals(individuals, permanent)
```

Arguments

`individuals`	An object of type Individual object. See details for description.
`permanent`	An object of type logical. Must be of length 1 (a singleton). The default value is `T`. See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 729.

Mark the individuals specified by individuals to be kept across tree sequence table simplification. This method may only be called if tree sequence recording has been turned on with initializeTreeSeq(). All currently living individuals are always kept across simplification; this method does not need to be called, and indeed should not be called, for that purpose. Instead, treeSeqRememberIndividuals() allows any individual, including dead individuals, to be kept in the final tree sequence. Typically this would be used, for example, to keep particular individuals that you wanted to be able to trace ancestry back to in later analysis. However, this is not the typical usage pattern for tree sequence recording; most models will

not need to call this method. There are two ways to keep individuals across simplification. If permanent is T (the default), then the specified individuals will be permanently remembered: their genomes will be added to the current sample, and they will always be present in the tree sequence. Permanently remembering a large number of individuals will, of course, markedly increase memory usage and runtime. Supplying F for permanent will instead mark the individuals only for (temporary) retention: their genomes will not be added to the sample, and they will appear in the final tree sequence only if one of their genomes is retained across simplification. In other words, the rule of thumb for retained individuals is simple: if a genome is kept by simplification, the genome's corresponding individual is kept also, if it is retained. Note that permanent remembering takes priority; calling this function with permanent=F on an individual that has previously been permanently remembered will not remove it from the sample. The behavior of simplification for individuals retained with permanent=F depends upon the value of the retainCoalescentOnly flag passed to initializeTreeSeq(); here we will discuss the behavior of that flag in detail. First of all, genomes are always removed by simplification unless they are (a) part of the final generation (i.e., in a living individual when simplification occurs), (b) ancestral to the final generation, (c) a genome of a permanently remembered individual, or (d) ancestral to a permanently remembered individual. If retainCoalescentOnly is T (the default), they are also always removed if they are not a branch point (i.e., a coalescent node or most recent common ancestor) in the tree sequence. In some cases it may be useful to retain a genome and its associated individual when it is simply an intermediate node in the ancestry (i.e., in the middle of a branch). This can be enabled by setting retainCoalescentOnly to F in your call to initializeTreeSeq(). In this case, ancestral genomes that are intermediate ("unary nodes", in tskit parlance) and are within an individual that has been retained using the permanent=F flag here are kept, along with the retained individual itself. Since setting retainCoalescentOnly to F will prevent the unary nodes for retained individuals from being pruned, simplification may often be unable to prune very much at all from the tree sequence, and memory usage and runtime may increase rapidly. If you are retaining many individuals, this setting should therefore be used only with caution; it is not necessary if you are purely interested in the most recent common ancestors. See the pyslim documentation for further discussion of retaining and remembering individuals and the effects of the retainCoalescentOnly flag. The metadata (age, location, etc) that are stored in the resulting tree sequence are those values present at either (a) the final generation, if the individual is alive when the tree sequence is output, or (b) the last time that the individual was remembered, if not. Calling treeSeqRememberIndividuals() on an individual that is already remembered will cause the archived information about the remembered individual to be updated to reflect the individual's current state; care should be taken to remember individuals at a point in time when their state is valid. A case where this is particularly important is for the spatial location of individuals in continuous-space models. SLiM automatically retains the portions of the genomes that comprise the first generation of any new subpopulation created with addSubpop() that are inherited by extant individuals, for easy recapitation and other analysis (see sections 17.2 and 17.10). However, the individuals of the first generation are not remembered automatically, only their needed genomic information.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Species: Sp, addSubpopSplit(), addSubpop(), countOfMutationsOfType(), individualsWithPedigreeID killIndividuals(), mutationCounts(), mutationFrequencies(), mutationsOfType(), outputFixedMutations(), outputFull(), outputMutations(), readFromPopulationFile(), recalculateFitness(), registerFitnessEffectCallback(), registerMateChoiceCallback(), registerModifyChildCallback(), registerMutationCallback(), registerMutationEffectCallback(), registerRecombinationCallback(), registerReproductionCallback(), registerSurvivalCallback(), simulationFinished(), skipTick(), subsetMutations(), treeSeqCoalesced(), treeSeqOutput(), treeSeqSimplify()

`treeSeqSimplify` *SLiM method treeSeqSimplify*

Description

Documentation for SLiM function `treeSeqSimplify`, which is a method of the SLiM class Species. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
treeSeqSimplify(void)
```

Arguments

`void`	An object of type . See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 730.

Triggers an immediate simplification of the tree sequence recording tables. This method may only be called if tree sequence recording has been turned on with initializeTreeSeq(). A call to this method will free up memory being used by entries that are no longer in the ancestral

path of any individual within the current sample (currently living individuals, in other words, plus those explicitly added to the sample with treeSeqRememberIndividuals()), but it can also take a significant amount of time. Typically calling this method is not necessary; the automatic simplification performed occasionally by SLiM should be sufficient for most models.

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Species: Sp, addSubpopSplit(), addSubpop(), countOfMutationsOfType(), individualsWithPedigreeID killIndividuals(), mutationCounts(), mutationFrequencies(), mutationsOfType(), outputFixedMutations(), outputFull(), outputMutations(), readFromPopulationFile(), recalculateFitness(), registerFitnessEffectCallback(), registerMateChoiceCallback(), registerModifyChildCallback(), registerMutationCallback(), registerMutationEffectCallback(), registerRecombinationCallback(), registerReproductionCallback(), registerSurvivalCallback(), simulationFinished(), skipTick(), subsetMutations(), treeSeqCoalesced(), treeSeqOutput(), treeSeqRememberIndividuals()

---

unevaluate *SLiM method unevaluate*

Description

Documentation for SLiM function `unevaluate`, which is a method of the SLiM class InteractionType. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a slim_block function further nested in a slim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
unevaluate(void)
```

Arguments

`void`	An object of type . See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 699.

Discards all evaluation of this interaction, for all subpopulations. The state of the InteractionType is reset to a state prior to evaluation. This can be useful if the model state has changed in such a way that the evaluation already conducted is no longer valid. For example, if the maximum distance, the interaction function, or the receiver or exerter constraints of the InteractionType need to be changed with immediate effect, or if the data used by an interaction() callback has changed in such a way that previously calculated interaction strengths are no longer correct, unevaluate() allows the interaction to begin again from scratch. In WF models, all interactions are automatically reset to an unevaluated state at the moment when the new offspring generation becomes the parental generation (at step 4 in the tick cycle; see section 23.4). In nonWF models, all interactions are automatically reset to an unevaluated state twice per tick: immediately after reproduction() callbacks have completed (after step 1 in the tick cycle; see section 24.1), and immediately before viability/survival selection (before step 4 in the tick cycle; see section 24.4). Given this automatic invalidation, most simulations have no reason to call unevaluate().

Value

An object of type void.

Copyright

Author(s)

Benjamin C Haller (`<bhaller@benhaller.com>`) and Philipp W Messer (`<messer@cornell.edu>`)

See Also

Other InteractionType: `IT`, `clippedIntegral()`, `distanceFromPoint()`, `distance()`, `drawByStrength()`, `evaluate()`, `interactingNeighborCount()`, `interactionDistance()`, `localPopulationDensity()`, `nearestInteractingNeighbors()`, `nearestNeighborsOfPoint()`, `nearestNeighbors()`, `neighborCountOfPoint()`, `neighborCount()`, `setConstraints()`, `setInteractionFunction()`, `strength()`, `testConstraints()`, `totalOfNeighborStrengths()`

`uniqueMutationsOfType`

SLiM method uniqueMutationsOfType

Description

Documentation for SLiM function `uniqueMutationsOfType`, which is a method of the SLiM class `Individual`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
uniqueMutationsOfType(mutType)
```

Arguments

`mutType`	An object of type integer or MutationType object. Must be of length 1 (a singleton). See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 685.

Returns an object vector of all the mutations that are of the type specified by mutType, out of all of the mutations in the individual. Mutations present in both genomes will occur only once in the result of this method, and the mutations will be given in sorted order by position, so this method is similar to sortBy(unique(individual.genomes.mutationsOfType(mutType)), "position"). It is not identical to that call, only because if multiple mutations exist at the exact same position, they may be sorted differently by this method than they would be by sortBy(). If you just need a count of the matching Mutation objects, rather than a vector of the matches, use -countOfMutationsOfType(). This method is provided for speed; it is much faster than the corresponding Eidos code. Indeed, it is faster than just individual.genomes.mutationsOfType(mutType), and gives uniquing and sorting on top of that, so it is advantageous unless duplicate entries for homozygous mutations are actually needed.

Value

An object of type Mutation object.

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Individual: `In`, `containsMutations()`, `countOfMutationsOfType()`, `relatedness()`, `setSpatialPosition()`, `sharedParentCount()`, `sumOfMutationsOfType()`

`usage` *SLiM method usage*

Description

Documentation for SLiM function `usage`, which is a method of the SLiM class `Community`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
usage(void)
```

Arguments

`void`	An object of type . See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 669.

Return the current memory usage of the simulation. The specifics of what is totalled up should not be relied upon as it may change from version to version of SLiM. This method is primarily useful for understanding where the memory usage of a simulation predominantly resides, for debugging or optimization. Note that it does not capture all memory usage by the process; rather, it summarizes the memory usage by SLiM and Eidos in directly allocated objects and buffers. To see details of this internal memory usage, use the Community method outputUsage(). To get the total memory usage of the running process (either current or peak), use the Eidos function usage().

Value

An object of type float. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other Community: Co, createLogFile(), deregisterScriptBlock(), genomicElementTypesWithIDs(), interactionTypesWithIDs(), mutationTypesWithIDs(), outputUsage(), registerEarlyEvent(), registerFirstEvent(), registerInteractionCallback(), registerLateEvent(), rescheduleScriptBlock(), scriptBlocksWithIDs(), simulationFinished(), speciesWithIDs(), subpopulationsWithIDs()

---

willAutolog *SLiM method willAutolog*

Description

Documentation for SLiM function `willAutolog`, which is a method of the SLiM class `LogFile`. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a `slim_block` function further nested in a `slim_script` function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
willAutolog(void)
```

Arguments

`void`	An object of type . See details for description.

Details

Documentation for this function can be found in the official SLiM manual: page 703.

Returns T if the log file is configured to log a new row automatically at the end of the current tick; otherwise, returns F. This is useful for calculating a value that will be logged only in ticks when the value is needed.

Value

An object of type void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or void or logical. Return will be of length 1 (a singleton)

Copyright

Author(s)

Benjamin C Haller (<bhaller@benhaller.com>) and Philipp W Messer (<messer@cornell.edu>)

See Also

Other LogFile: LF, addCustomColumn(), addCycleStage(), addCycle(), addKeysAndValuesFrom(), addMeanSDColumns(), addPopulationSexRatio(), addPopulationSize(), addSubpopulationSexRatio(), addSubpopulationSize(), addSuppliedColumn(), addTick(), clearKeysAndValues(), flush(), logRow(), setFilePath(), setLogInterval(), setSuppliedValue(), setValue()

%.% *slimrlang stub for the SLiM '.' operator*

Description

Use this in place of '.' from SLiM to specify a method or a property coming from a particular SLiM class. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a \code\linkslim_block function further nested in a \code\linkslim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
lhs %.% rhs
```

Arguments

lhs	Object of class rhs, to extract methods or properties from
rhs	SLiM class R object (such as Subpopulation, .M, etc.). Type slim_classes for a table of possible values.

Copyright

Author(s)

Benjamin C Haller (\emailbhaller@benhaller.com) and Philipp W Messer (\emailmesser@cornell.edu)

`%else%` *slimrlang stub for the second part of the SLiM ternary operator (*`condition ? yes else no`*).*

Description

This is used in conjunction with which will make the code valid in R. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a \code\linkslim_block function further nested in a \code\linkslim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
lhs %else% rhs
```

Arguments

`lhs`	A SLiM expression to be executed if the condition (before the companion is `TRUE`
`rhs`	A SLiM expression to be executed if the condition (before the companion is `FALSE`

Copyright

Author(s)

Benjamin C Haller (\emailbhaller@benhaller.com) and Philipp W Messer (\emailmesser@cornell.edu)

`%?%` *slimrlang stub for the first part of the SLiM ternary operator (*`condition ? yes else no`*).*

Description

This is used in conjunction with which will make the code valid in R. Note that the R function is a stub, it does not do anything in R (except bring up this documentation). It will only do anything useful when used inside a \code\linkslim_block function further nested in a \code\linkslim_script function call, where it will be translated into valid SLiM code as part of a full SLiM script.

Usage

```
lhs %?% rhs
```

Arguments

`lhs`	A condition
`rhs`	A SLiM expression to be executed if the condition is `TRUE`

Copyright

Author(s)

Benjamin C Haller (\emailbhaller@benhaller.com) and Philipp W Messer (\emailmesser@cornell.edu)

Index