Package: fibre (via r-universe)

September 2, 2024

Title Fast Evolutionary Trait Modelling on Phylogenies using Branch Regression Models

Version 0.0.0.9000

Description Implements Phylogenetic Branch Regression models which allow for flexible and versatile models of evolution along a phylogeny. The model can be used to detect shifts in rates of evolution along branches. The model uses a continuous and linear model structure and so can be easily combined with other non-phylogenetic statistical structures, as long as they are implemented using the R package INLA. One major uses of this are to condition on phylogeny in a standard regression between two traits, thus 'accounting' for phylogenetic structure in the response variable, similar to how pgls is used but allowing for a more flexible phylogenetic model. This also allows the phylogenetic model to be combined with the spatial models that INLA excels at (and with comparable flexibility to those spatial models).

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Suggests knitr, rmarkdown, bench, RPANDA, testthat (>= 3.0.0), INLA, tidytree, recipes, learnr, roxygen2, rgl, patchwork, rmarchingcubes, Rvcg, orientlib, Morpho, einsum, imager, movMF, furrr, ggridges, ggforce

VignetteBuilder knitr

Imports ape, dplyr, igraph, magrittr, fastmatch, MCMCglmm, phytools, phangorn, ivs, phyf (>= 0.0.0.9000), hardhat, rlang (>= 0.4.11), stats, vctrs, zeallot, torch, glue, Matrix, generics, ggplot2, glmnet, purrr, skimr, cli, tidyr, methods, utils

Config/testthat/edition 3

Depends R (>= 2.10)

Remotes rdinnager/phyf

URL https://rdinnager.github.io/fibre/

Additional_repositories https://inla.r-inla-download.org/R/stable

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

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

RemoteRef HEAD

RemoteSha f62fd4a6f0cc8d9492d909115582cb3ae21bbe18

Contents

`bre` *Specify a branch length (random) effect*

Description

This function is meant to be called only in the `formula` argument of `fibre()`.

Usage

```
bre(
  phyf,
 rate_distribution = c("iid", "laplacian", "student-t", "horseshoe", "Brownian", "re"),
  hyper = list(prec = list(prior = "pc.prec", param = c(1, 0.1))),
  latent = 0,
  label = NULL,
  standardise = TRUE
)
```

Arguments

phyf	A pfc column containing the phylogenetic structure
rate_distribution	What distribution to use to model rates of evolution?
hyper	Hyper parameters as a list. Specify the prior distribution for engine = INLA models here. Default is a penalised complexity prior with 10% prior probability density greater than 1, which tend to work well for standardised Gaussian responses and Binomial responses.
latent	How many latent variables to generate in engine = INLA models. Default is none.
label	An optional label used to identify the random effect later The default is a label generated from the expression in phyf
standardise	Should the pfc object be standardised based on it's implied typical variance for terminal nodes? Default: TRUE. This helps random effects to be comparable to each other.

Value

A list of data to be used by the model.

---

bre_brownian *Specify a branch length (random) effect for a Brownian motion model*

Description

This function is meant to be called only in the formula argument of fibre().

Usage

```
bre_brownian(
  phyf,
  hyper = list(prec = list(prior = "pc.prec", param = c(1, 0.1))),
  latent = 0,
  label = NULL,
  standardise = TRUE
)
```

Arguments

phyf	A pfc column containing the phylogenetic structure
hyper	Hyper parameters as a list. Specify the prior distribution for engine = INLA models here. Default is a penalised complexity prior with 10% prior probability density greater than 1, which tend to work well for standardised Gaussian responses and Binomial responses.

latent	How many latent variables to generate in engine = INLA models. Default is none.
label	An optional label used to identify the random effect later The default is a label generated from the expression in phyf
standardise	Should the pfc object be standardised based on it's implied typical variance for terminal nodes? Default: TRUE. This helps random effects to be comparable to each other.

Value

A list of data to be used by the model.

`bre_second_order` *Specify a branch length (random) effect for a 'Second Order' Brownian motion model*

Description

This function is meant to be called only in the formula argument of fibre().

Usage

```
bre_second_order(
  phyf,
  hyper = list(prec = list(prior = "pc.prec", param = c(1, 0.1))),
  latent = 0,
  label = NULL,
  standardise = TRUE
)
```

Arguments

phyf	A pfc column containing the phylogenetic structure
hyper	Hyper parameters as a list. Specify the prior distribution for engine = INLA models here. Default is a penalised complexity prior with 10% prior probability density greater than 1, which tend to work well for standardised Gaussian responses and Binomial responses.
latent	How many latent variables to generate in engine = INLA models. Default is none.
label	An optional label used to identify the random effect later The default is a label generated from the expression in phyf
standardise	Should the pfc object be standardised based on it's implied typical variance for terminal nodes? Default: TRUE. This helps random effects to be comparable to each other.

Value

A list of data to be used by the model.

evo_autodecoder *Create a Evolutionary Autodecoder Model*

Description

Create a Evolutionary Autodecoder Model

Usage

```
evo_autodecoder(
  latent_dim,
  n_edges,
  decoder,
  reconstruction_loss,
  device,
  decoder_args = list(),
  loss_args = list()
)
```

Arguments

latent_dim	Number of latent dimensions
decoder	A torch::nn_model() specifying a 'decoder' network architecture. The decoder network should accept a 2 dimensional torch::torch_tensor() with first d

Value

A torch::nn_module()

fibre *Fit a* fibre

Description

fibre() fits a model.

Usage

```
fibre(x, ...)

## Default S3 method:
fibre(x, ...)

## S3 method for class 'data.frame'
```

```
fibre(
  x,
  y,
  intercept = TRUE,
  engine = c("inla", "glmnet", "torch"),
  engine_options = list(),
  ncores = NULL,
  verbose = 0,
  ...
)

## S3 method for class 'matrix'
fibre(
  x,
  y,
  intercept = TRUE,
  engine = c("inla", "glmnet", "torch"),
  engine_options = list(),
  ncores = NULL,
  verbose = 0,
  ...
)

## S3 method for class 'formula'
fibre(
  formula,
  data,
  intercept = TRUE,
  family = "gaussian",
  engine = c("inla", "glmnet", "torch"),
  engine_options = list(),
  ncores = NULL,
  verbose = 0,
  ...
)

## S3 method for class 'recipe'
fibre(
  x,
  data,
  intercept = TRUE,
  engine = c("inla", "glmnet", "torch"),
  engine_options = list(),
  ncores = NULL,
  verbose = 0,
  ...
)
```

Arguments

x	Depending on the context: • A **data frame** of predictors. • A **matrix** of predictors. • A **recipe** specifying a set of preprocessing steps created from `recipes::recipe()`.
...	Not currently used, but required for extensibility.
y	When x is a **data frame** or **matrix**, y is the outcome specified as: • A **data frame** with 1 numeric column. • A **matrix** with 1 numeric column. • A numeric **vector**.
formula	A formula specifying the outcome terms on the left-hand side, and the predictor terms on the right-hand side.
data	When a **recipe** or **formula** is used, data is specified as: • A **data frame** containing both the predictors and the outcome.

Value

A `fibre` object.

Examples

```
predictors <- mtcars[, -1]
outcome <- mtcars[, 1]

# XY interface
#mod <- fibre(predictors, outcome)

# Formula interface
#mod2 <- fibre(mpg ~ ., mtcars)

# Recipes interface
#library(recipes)
#rec <- recipe(mpg ~ ., mtcars)
#rec <- step_log(rec, disp)
#mod3 <- fibre(rec, mtcars)
```

---

get_aces *Title*

Description

Title

Usage

```
get_aces(
  x,
 type = c("marginals", "samples", "mode", "mean", "median", "lower", "upper", "ci",
    "hpd", "sd"),
  n = 1,
  p = 0.05
)
```

Arguments

x	A fitted model object produced by fibrer
type	What kind of posterior summary to return?
n	If type = "samples", how many samples to return?
p	If type = "hpd", what alpha levels to use?

Value

For all types except "hpd", "ci", and "marginals", a numeric vector, otherwise a list for "hpd" and "marginals", and a matrix for "ci".

---

get_rates *Title*

Description

Title

Usage

```
get_rates(
  x,
 type = c("marginals", "samples", "mode", "mean", "median", "lower", "upper", "ci",
    "hpd", "sd"),
  n = 1,
  p = 0.05
)
```

Arguments

x	A fitted model object produced by fibrer
type	What kind of posterior summary to return?
n	If type = "samples", how many samples to return?
p	If type = "hpd", what alpha levels to use?

Value

For all types except "hpd", "ci", and "marginals", a numeric vector, otherwise a list for "hpd" and "marginals", and a matrix for "ci".

`get_tces` *Title*

Description

Title

Usage

```
get_tces(
  x,
  type = c("marginals", "samples", "mode", "mean", "median", "lower", "upper", "ci",
    "hpd", "sd"),
  n = 1,
  p = 0.05
)
```

Arguments

`x`	A fitted model object produced by `fibrer`
`type`	What kind of posterior summary to return?
`n`	If `type = "samples"`, how many samples to return?
`p`	If `type = "hpd"`, what alpha levels to use?

Value

For all types except "hpd", "ci", and "marginals", a numeric vector, otherwise a list for "hpd" and "marginals", and a matrix for "ci".

`get_tips` *Title*

Description

Title

Usage

```
get_tips(
  x,
 type = c("marginals", "samples", "mode", "mean", "median", "lower", "upper", "ci",
    "hpd", "sd"),
  n = 1,
  p = 0.05
)
```

Arguments

x	A fitted model object produced by fibrer
type	What kind of posterior summary to return?
n	If type = "samples", how many samples to return?
p	If type = "hpd", what alpha levels to use?

Value

For all types except "hpd", "ci", and "marginals", a numeric vector, otherwise a list for "hpd" and "marginals", and a matrix for "ci".

---

load_model *Load a model*

Description

Load a model

Usage

```
load_model(name)
```

Arguments

name	Name of the model. Currently only "bird_beak".

Value

A torch::nn_module() with pre-trained weights

Examples

```
if(torch::torch_is_installed()) {
model <- load_model("bird_beaks")
}
```

predict.fibre *Predict from a* fibre

Description

Predict from a fibre

Usage

```
## S3 method for class 'fibre'
predict(object, new_data = NULL, type = "numeric", ...)
```

Arguments

object	A fibre object.
new_data	A data frame or matrix of new predictors.
type	A single character. The type of predictions to generate. Valid options are: • "numeric" for numeric predictions.
...	Not used, but required for extensibility.

Value

A tibble of predictions. The number of rows in the tibble is guaranteed to be the same as the number of rows in new_data.

Examples

```
train <- mtcars[1:20,]
test <- mtcars[21:32, -1]

# Fit
#mod <- fibre(mpg ~ cyl + log(drat), train)

# Predict, with preprocessing
#predict(mod, test)
```

re — *Specify a random effect*

Description

This function is meant to be called only in the `formula` argument of `fibre()`.

Usage

```
re(
  groups,
  hyper = list(prec = list(prior = "pc.prec", param = c(1, 0.1))),
  label = NULL,
  standardise = TRUE
)
```

Arguments

`groups`	A character or factor column containing the grouping variable for the random effect
`hyper`	Hyper parameters as a list. Specify the prior distribution for `engine = INLA` models here. Default is a penalised complexity prior with 10% prior probability density greater than 1, which tend to work well for standardised Gaussian responses and Binomial responses.
`label`	An optional label used to identify the random effect later The default is a label generated from the expression in `phyf`
`standardise`	Should the `pfc` object be standardised based on it's implied typical variance for terminal nodes? Default: `TRUE`. This helps random effects to be comparable to each other.

Value

A list of data to be used by the model.

sdf_net — *A signed distance field based neural network model for generating 3d shapes*

Description

A signed distance field based neural network model for generating 3d shapes

Usage

```
sdf_net(n_latent = 64, breadth = 512)
```

Arguments

n_latent	Number of dimensions for the latent space
breadth	Breadth of the multilayer perceptron networks

Value

A torch::nn_module()

Examples

```
if(torch::torch_is_installed()) {
sdf_net()
}
```

---

simulate_traits *Function to simulate continuous trait value histories on a phylogeny.*

Description

Function to simulate continuous trait value histories on a phylogeny.

Usage

```
simulate_traits(
  phy,
  rate_model = c("continuous", "discrete"),
  temp_trend_rates = 0,
  rate_change,
  rates = NULL,
  anc = c(`1` = 0),
  internal = FALSE,
  nsim = 1,
  pos_strat = c("none", "log", "add_const"),
  temp_trend_mean = 0
)
```

Arguments

phy	A phylogenetic tree (phylo object) on which to simulate traits
rate_model	The type of rate model for how rates of evolution evolve on the phylogeny: "continuous" for continuous Brownian motion evolution of rates, or "discrete" for evolution of rate "classes" across the phylogeny, using an mk model.
temp_trend_rates	What temporal trend in rates should there be? A positive number for an increase, and negative number for a decrease with the magnitude controlling the strength of linear change. This trend is added to rates simulated under the rate_model.

rate_change — If rate_model is "continuous", this should be a single positive number controlling how fast rates change continuously along the tree. If rate_model is "discrete", this should be a transition matrix for the rate classes. Or, if rate_model is "discrete", and this can be a length 2 numeric vector specifying

rates — Only used if rate_model is "discrete", in which case this should be a named vector whose values are the rates in each rate class, and whose names are the rate class states (e.g. c("1" = 3, "2" = 10)). See sim.history, for more detail on how the discrete model works. Or, if an unnamed numeric vector of length two, a mean and standard deviation parameterizing a normal distribution from which to draw rates for each rate class. If NULL, rates will be drawn from a normals distribution with mean = 0 and sd = 1.

anc — Value of the trait at the root ancestor. For rate_model = "discrete", can be a length 1 named vector where the name is the ancestral state, and the value is the trait starting value. For rate_model = "continuous", any names are ignored, but should be length 2, where the first element is the ancestral trait value and the second element is the ancestral rate of evolution.

internal — Logical value. If TRUE return trait states at internal nodes.

nsim — Number of simulation to run.

pos_strat — ?

temp_trend_mean — A temporal trend in rates.

Value

A vector or matrix (for nsim > 1) containing simulated trait values for each tip if internal = FALSE, or for each node if internal = TRUE

---

tidy.fibre *Tidy Model Results*

Description

Tidy Model Results

Usage

```
## S3 method for class 'fibre'
tidy(
  x,
  effects = c("fixed", "rates", "random", "hyper"),
  conf.type = c("cred.int", "marginals"),
  indexes = NULL,
  ...
)
```

Arguments

x	A fibre model object
effects	Which effects do you want tidied? One of: "fixed", for fixed effects, "random" for random effects, or "hyper" for the hyper-parameters of the random effects. Can also be "rates", which is a synonym for "random", since the random effects are rates of trait evolution along phylogenetic edges.
conf.type	What kind of confidence interval. Choices are: "cred.int" for approximate Bayesian marginal credible intervals. or "marginals" for the full approximate marginal distributions, as a data.frame with value and y.value columns. value is the value of the parameter, and y.value is the marginal posterior density (e.g. what value is the x axis and y.value is the y axis when plotting the posterior density).
indexes	If effects = "random" or effects = "rates", this is a vector of indices to retrieve particular random effects. Default is to return all random effects, however, this can be slow for retrieving the marginals.
...	Not used.

Value

A tidy tibble with information about the fitted model parameters.

Index