Package ‘phylopairs’

April 23, 2025

Title Comparative Analyses of Lineage-Pair Traits

Version 0.1.1

Description Facilitates the testing of causal relationships among lineage-pair traits in a phylogenetically informed context. Lineage-pair traits are characters that are defined for pairs of lineages instead of individual taxa. Examples include the strength of reproductive isolation, range overlap, competition coefficient, diet niche similarity, and relative hybrid fitness. Users supply a lineage-pair dataset and a phylogeny. 'phylopairs' calculates a covariance matrix for the pairwise-defined data and provides built-in models to test for relationships among variables while taking this covariance into account. Bayesian sampling is run through built-in 'Stan' programs via the 'rstan' package. The various models and methods that this package makes available are described in Anderson et al. (In Review), Coyne and Orr (1989) <doi:10.1111/j.1558-5646.1989.tb04233.x>, Fitzpatrick (2002) <doi:10.1111/j.0014-3820.2002.tb00860.x>, and Castillo (2007) <doi:10.1002/ece3.3093>.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.2

Biarch true

Depends R (>= 3.5.0)

Imports ape, loo, methods, phytools, Rcpp (>= 0.12.0), RcppParallel (>= 5.0.1), rstan (>= 2.18.1), rstantools (>= 2.4.0)

LinkingTo BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1), rstan (>= 2.18.1), StanHeaders (>= 2.18.0)

SystemRequirements GNU make

NeedsCompilation yes

Author Sean A. S. Anderson [aut, cre]

Maintainer Sean A. S. Anderson <sean.as.anderson@gmail.com>

Repository CRAN

Date/Publication 2025-04-23 20:30:01 UTC

Contents

---

phylopairs-package *The 'phylopairs' package.*

Description

phylopairs provides tools for conducting comparative analyses of lineage-pair traits in a phylogenetically informed context. Regression-style analsyses of pairwise-defined traits like "strength of RI" or "diet niche overlap" collected for numerous pairs of related taxa can yield important insights to biologists, but it is important to recognize that such data are not independent. phylopairs provides a function, `taxapair.vcv()`, that calculates the expected covariance structure of a pairwise-defined trait given (1) a table of pairs, (2) a phylogeny, and (3) a chosen model (as described in Anderson et al. *in review*). This covariance structure can be used in any number of downstream analyses. phylopairs provides tools for a few such analyses, including linear regression, linear mixed models, generalized least squares, and two forms of beta regression for use with bounded response variables. All analyses are conducted in a Bayesian framework using built-in Stan software programs interfaced through the rstan package.

Author(s)

Maintainer: Sean A. S. Anderson <sean.as.anderson@gmail.com>

References

Stan Development Team (NA). RStan: the R interface to Stan. R package version 2.32.6. https://mc-stan.org

Anderson et al. *in review*. The comparative analysis of lineage-pair data.

betareg.stan *betareg.stan*

Description

Fits one of two beta regression models to a dataset in a Bayesian framework using Stan software via the rstan package. Response variable must be bounded between 0 and 1. Users can fit either (1) a basic beta regression model or a (2) beta mixed-effects model in which there are covarying random intercepts in the linear predictor. For the latter, users must supply a covariance matrix. In both models, users can choose one of four link functions. Users can alter parameters for model-parameter prior distributions and Bayesian sampling settings. See details.

Usage

```
betareg.stan(des, y, model="beta.reg", link="logit", covmat=NULL,
  iter=2000, chains=4, coef.u=0, coef.sd=10, phi.shape=0.01, phi.rate=0.01,
  physig2.u=-1, physig2.sd=1, cores=4, ...)
```

Arguments

des	A vector of predictor variable observations OR, in the case of multiple predictors, a matrix in which each column is a vector of observations of a given predictor. betareg.stan() adds a column of 1s to make this a design matrix whose first column corresponds to the model intercept (unless such a column already exists).
y	A vector of response variable observations.
model	One of "beta.reg" or "beta.mm" for a basic beta regression model or beta mixed model, respectively; defaults to "beta.reg".
link	The link function to be used. Default is "logit". Other possible choices are "probit", "cloglog" (complementary log-log), and "loglog".
covmat	Covariance matrix for model residuals (a lineage-pair covariance matrix if analyzing lineage-pair data or a phylogenetic vcv matrix if analyzing bounded species data).
iter	Number of iterations to run on each chain; defaults to 2000 (more are often necessary).
chains	Number of Markov chains to run; defaults to 4.
coef.u	Mean of the Gaussian prior for each preditor variable coefficient; defaults to 0.
coef.sd	SD of the Gaussian prior for each preditor variable coefficient; defaults to 10.
phi.shape	Shape parameter for gamma prior of beta distribution's precision parameter (phi); defaults to 0.01.
phi.rate	Rate parameter for gamma prior of beta distribution's precision parameter (phi); defaults to 0.01.
physig2.u	Mean of the lognormal prior for the scale of the residual covariance; defaults to -1.

`physig2.sd` SD of the lognormal prior for the scale of the residual covariance; defaults to 1.

`cores` Number of cores to be used; defaults to 4 (one chain per core for most laptops).

`...` additional arguments passed to `rstan::sampling`, including control parmeters (see rstan::sampling documentation)

Details

The two models that can be chosen differ in terms of their linear predictor on the scale of the link function. These predictors are

1. Standard beta regression: `g(u) = XB`
2. Beta mixed-effects regression: `g(u) = XB + u, u~N(0, physig2*C)`, where C is a lineage-pair covariance matrix (if analyzing lineage-pair data) or a phylogenetic vcv matrix (if analyzing species data) and physig2 is the scaling parameter for C.

NOTE: For model 2, the user must supply a covariance matrix. These are generated with the 'taxapair.vcv' function.

Prior Distributions for Model Parameters: The underlying stan models assume the following prior distributions for model parameters.

1. Regression coefficients: Gaussian prior (users can set prior mean and sd).
2. Precision parameter `phi`: gamma prior (users can set prior shape and rate of gamma).
3. `physig`: lognormal prior (users can set prior mean and sd).

Value

A list containing two elements: (1) the posterior distribution of model parameters, and (2) the log-likelihood of the posteriors for use in downstream analyses (e.g. the calculation of model fitting metrics like loo or waic). For interpreting model parameters, note that `Coef[1]` is the intercept and `Coef[2]`, `Coef[3]`, ... , `Coef[N]` are regression coefficient for the 1st-(N-1)th predictor variables.

References

Anderson, S. A. S., et al. *In review*. The comparative analysis of lineage-pair data.

Examples

```
## Example 1: Fit beta regression models with different link functions to independent data
# Load a dataset of independent response observations simulated with a logit link function
data(data5)
# Note: data were simulated with Coef[1]=1 (intercept), Coef[2]=0.8 (slope), phi=5

# Run the betareg function
result1 = betareg.stan(des=data5[,3], y=data5[,4], iter=1000, cores=2)

# Fit the model again but this time use a probit link function
result2 = betareg.stan(des=data5[,3], y=data5[,4], link="probit", iter=1000, cores=2)

# Compare posterior parameter estimates
result1[[1]]
```

```
result2[[1]]

## Example 2: Fit beta regression models to a dataset with simulated non-independence
# Load a dataset of non-independent response observations simulated with a logit link function
data(data7)
# Note: data were simulated with Coef[1]=1 (intercept), Coef[2]=0.8 (slope), phi=5
# Load the lineage-pair covariance matrix that arose from those simulations
data(sim.cov.pairs)

# Run the betareg function
result1 = betareg.stan(des=data7[,3], y=data7[,4], model="beta.mm", cov=sim.cov.pairs,
 iter=1000, cores=2)

# Fit the model again but this time without the covariance matrix
result2 = betareg.stan(des=data7[,3], y=data7[,4], iter=1000, cores=2)

# Fit the model a third time with the cov. matrix but now with a probit link function
result3 = betareg.stan(des=data7[,3], y=data7[,4], model="beta.mm",
  cov=sim.cov.pairs, link="probit", iter=1000, cores=2)

# Compare posterior parameter estimates
result1[[1]]
result2[[1]]
result3[[1]]

# Compare the fit of the three models via leave-on-out (loo) cross validation.
loo1 = suppressWarnings(loo::loo(result1[[2]]))
loo2 = suppressWarnings(loo::loo(result2[[2]]))
loo3 = suppressWarnings(loo::loo(result3[[2]]))
loo::loo_compare(loo1, loo2, loo3)
```

covmat.check *covmat.check*

Description

Tets for validity of a covariance matrix based on four conditions: symmetry, diagonal dominance, positive definiteness, and positive variance.

Usage

```
covmat.check(mat)
```

Arguments

mat	A putative covariance matrix.

Details

A valid covariance matrix must be symmetric, diagonally dominant (largest values in each row are on the diagonal), positive definite, and have positive variance. `covmat.check` takes a matrix as input and tests for these four conditions.

Value

A data.frame containing logical "TRUE" or "FALSE" for each condition.

Examples

```
# Load sample covariance matrix
data(sim.cov.pairs)
# Test for validity
covmat.check(sim.cov.pairs)
```

`linreg.stan` *linreg.stan*

Description

Fits one of three models of linear regression to a dataset in a Bayesian framework. Bayesian sampling is conducted in the Stan software via the ’rstan’ package. Users supply vectors of observations and, unless an "ols" model is chosen, a covariance matrix. Users can alter parameters for model-parameter prior distributions and Bayesian sampling settings. See details.

Usage

```
linreg.stan(des, y, model=”ols”, covmat=NULL, iter=2000,
  chains=4, cores=4, coef.u=0, coef.sd=10, physig2.u=-1, physig2.sd=1,
  randsig2.loc=0, randsig2.sc=2.5, ...)
```

Arguments

`des` A vector of predictor variable observations OR, in the case of multiple predictors, a matrix in which each column is a vector of observations of a given predictor. Function will add a column of 1s to make this a design matrix whose first column corresponds to the model intercept (unless such a column already exists).

`y` A vector of response variable observations.

`model` Choice of "ols", "pgls", or "lp.gls"; defaults to ols.

`covmat` Covariance matrix for model residuals (a lineage-pair covariance matrix if analyzing lineage-pair data or a phylogenetic vcv matrix if analyzing species data).

`iter` Number of iterations to run on each chain; defaults to 2000 (more are often necessary).

`chains` Number of Markov chains to run; defaults to 4.

`cores`	Number of cores to use.
`coef.u`	Mean of the Gaussian prior for each preditor variable coefficient; defaults to 0.
`coef.sd`	SD of the Gaussian prior for each preditor variable coefficient; defaults to 10.
`physig2.u`	Mean of the prior distribution (lognormal) for the scale of the phylogenetic component of residual covariance; defaults to -1.
`physig2.sd`	SD of the prior distribution (lognormal) for the scale of the phylogenetic component of residual covariance; defaults to 1.
`randsig2.loc`	Location parameter of the prior distribution (cauchy) for the scale of the independent component of residual covariance ; defaults to 0.
`randsig2.sc`	Scale parameter of the prior distribution (cauchy) for the scale of the independent component of residual covariance ; defaults to 2.5.
`...`	additional arguments passed to `rstan::sampling()`, including control parmeters (see `rstan::sampling()` documentation)

Details

The models that can be chosen are:

1. "ols": basic ordinary least-squares regression.

   `Y` = `XB` + `epsilon`; where `epsilon~N(0,randsig2*I)` and randsig2 is the scaling parameter for identity matrix I.

2. "pgls": phylogenetic generalized least-squares regression in which residuals covary according to covariance matrix C. This is a lineage-pair covariance matrix (if analyzing lineage-pair data) or a phylogenetic vcv matrix (if analyzing species data).

   `Y` = `XB` + `epsilon`; where `epsilon~N(0,physig2*C)` and physig2 is the scaling parameter for covariance matrix C.

3. "lp.gls" a mixed-effects version of pgls in which the residuals have both uncorrelated and correlated components, where the latter are structured according to covariance matrix C. This is the 'lp.gls' model from Anderson et al. 2025 (In Revision).

   `Y` = `XB` + `epsilon` + `u`; where `epsilon~N(0,randsig2*I)` and `u~N(0,physig2*C)`.

NOTE: For models 2 and 3, the user must supply a covariance matrix.

Prior Distributions for Regression Model Parameters: The underlying stan models assume the following prior distributions for model parameters

1. Regression coefficients: Gaussian prior (users can set prior mean and sd).
2. `physig2`: lognormal prior (users can set prior mean and sd).
3. `randsig2`: cauchy prior (users can set location and scale parameters of prior).

Value

A list containing two elements: (1) the posterior distribution of model parameters, and (2) the log-likelihood of the posteriors for use in downstream analyses (e.g. the calculation of model fitting metrics like loo or waic). For interpreting model parameters, note that `Coef[1]` is the intercept and `Coef[2]`, `Coef[3]`, ... , `Coef[N]` are regression coefficient for the 1st-(N-1)th predictor variables.

References

Anderson, S. A. S., et al. *In review*. The comparative analysis of lineage-pair data.

Examples

```
## Fit regression models with and without covariance matrix
# Note: data were simulated with Coef[1]=1 (intercept), Coef[2]=0.8 (slope)
# Load a data simulated with a non-independent response observations
data(data3)
# Also load the lineage-pair covariance matrix that arose from those simulations
data(sim.cov.pairs)
# Fit an OLS model
result1 = linreg.stan(des=data3[,3], y=data3[,4], cores=2)
# Fit an pgls model
result2 = linreg.stan(des=data3[,3], y=data3[,4], model=”pgls”, covmat=sim.cov.pairs, cores=2)

# Compare posterior parameter estimates
result1[[1]]
result2[[1]]

# Compare the fit of the two models via loo and waic
loo1 = loo::loo(result1[[2]])
loo2 = loo::loo(result2[[2]])
waic1 = loo::waic(result1[[2]])
waic2 = loo::waic(result2[[2]])
loo1
loo2
waic1
waic2
loo::loo_compare(loo1, loo2)
loo::loo_compare(waic1, waic2)

# Extend the comparison by fitting a lp.gls model
result3 = linreg.stan(des=data3[,3], y=data3[,4], model=”lp.gls”,
 covmat=sim.cov.pairs, cores=2)

# Compare posterior parameter estimates
result1[[1]]
result2[[1]]
result3[[1]]

# Compare the fit of the three models via loo and waic
loo3 = loo::loo(result3[[2]])
waic3 = loo::waic(result3[[2]])
loo::loo_compare(loo1, loo2, loo3)
loo::loo_compare(waic1, waic2, waic3)
```

node.averager *node.averager*

Description

Calculate weighted or unweighted node-averaged values of a lineage-pair trait.

Usage

```
node.averager(dataset, tree, taxacolumns, varb,
  weighted=FALSE, prune=TRUE, av=TRUE)
```

Arguments

`dataset` A data.frame in which each row corresponds to a pair in the dataset. Must contain two columns of taxa names (one for each taxon in every pair) and at least one data column with the pairwise-defined trait that is to be averaged. Taxa names must be in same format as that used in the tree.

`tree` An ultrametric phylogenetic tree ('phylo' object) containing the species that appear in at least one pair in the dataset. Names must be in the same format as those used in 'dataset'.

`taxacolumns` Character vector containing the column names for the two columns containing species names (e.g. c("sp1", "sp2"))

`varb` The variable to be averaged (e.g. "RI", "range_overlap", etc.)

`weighted` Logical indicating whether weighted node averages are to be calculated; defaults to FALSE.

`prune` Logical indicating whether tree should be pruned to contain just the species represented in the dataset; defaults to TRUE.

`av` Logical indicating whether to average the values of multiple entries for the same pair, should they appear in the dataset; defaults to TRUE. If set to FALSE, function will stop in the case of more than one entry in the dataset corresponding to the same pair.

Details

`node.averager()` takes a lineage-pair dataset and a phylogenetic tree and returns the average value of the pairwise-defined trait at each node. It calculates, at each node in the tree, the average value of a pairwise-defined trait for all pairs whose species span the node.

The simple or 'unweighted' average is calculated as introduced by Coyne and Orr (1989). The 'weighted' node averaging procedure was introduced by Fitzpatrick (2002) and discussed in Fitzpatrick and Turelli (2006). In weighted averaging, the trait values for the pairs spanning a node are first halved K-1 times, where K is the number of nodes between the species in a pair. These halved values are then summed to get arrive at a weighted average for a node.

Note: For datasets containing many individual species, the best available tree might be very large. By default, `node.averager()` prunes the tree to contain just the species represented in the dataset. Beware that pruning can affect both the number of nodes and the node averaged values (for example by altering the number of nodes between pairs when calculating weighted node averages).

Value

Numeric vector of node averages, named according to node indices.

References

Coyne, J. A., Orr, H. A. 1989. Patterns of speciation in Drosophila. Evolution 43:362-381.

Fitzpatrick, B. M. 2002. Molecular correlates of reproductive isolation. Evolution 56:191-198.

Fitzpatrick, B. M., Turelli. 2006. The geography of mammalian speciation: mixed signals from phylogenies and range maps. Evolution 60:601-615.

Examples

```
# Load simulated dataset and tree
data(data1)
data(sim.tree1)

# Perform node averaging
unwtd <- node.averager(dataset = data1, tree = sim.tree1, varb = ”pred”,
  taxacolumns = c(”sp1”, ”sp2”))
wtd <- node.averager(dataset = data1, tree = sim.tree1, varb = ”pred”,
  taxacolumns = c(”sp1”, ”sp2”), weighted = TRUE)

# Compare outcomes of weighted and unweighted node averaging
unwtd
wtd
summary(unwtd)
summary(wtd)

# Calculate data loss
nrow(data1) - length(wtd)
nrow(data1) - length(unwtd)


# Plot tree and node labels
library(ape)
plot(sim.tree1)
nodelabels()
```

pair.age.in.tree *pair.age.in.tree*

Description

Takes a lineage-pair dataset and a phylogenetic tree and returns the ’age’ of the node representing the MRCA for each pair. ’Age’ is measured from the present and is in whatever units are represented by branch lengths of the tree (which is time in a dated phylogeny).

Takes a lineage-pair dataset and a phylogenetic tree and returns the ’age’ of the node representing the MRCA for each pair. ’Age’ is measured from the present and is in whatever units are represented by branch lengths of the tree (which is time in a dated phylogeny).

Usage

```
pair.age.in.tree(dataset, tree, taxacolumns)
```

```
pair.age.in.tree(dataset, tree, taxacolumns)
```

Arguments

`dataset` A data.frame in which each row corresponds to a pair in the dataset. Must contain two columns of taxa names (one for each taxon in every pair), and the taxa names must be in same format as that used in the tree.

`tree` An ultrametric phylogenetic tree ('phylo' object) containing the species that appear in at least one pair in the dataset. Names must be in the same format as those used in 'dataset'.

`taxacolumns` Character vector containing the column names for the two columns containing species names (e.g. c("sp1", "sp2"))

Details

Function is a wrapper for `ape::node.depth.edgelength`. That function returns the depths of all the nodes in a tree as measured from the root. `phylopairs::node.age` converts these values to depth as measured from the present and returns the values corresponding to each pair in a user-supplied lineage-pair dataset.

Function is a wrapper for `ape::node.depth.edgelength()`. That function returns the depths of all the nodes in a tree as measured from the root. `phylopairs::node.age()` converts these values to depth as measured from the present and returns the values corresponding to each pair in a user-supplied lineage-pair dataset.

Value

A numeric vector of node ages ordered to match the rows in 'dataset'.

A numeric vector of node ages ordered to match the rows in 'dataset'.

Examples

```
# Load a dataset and a tree
data(data1)
data(sim.tree1)

# Find the node.ages and add as a column to the dataset
data1$age = pair.age.in.tree(dataset=data1, tree=sim.tree1, taxacolumns=c("sp1","sp2"))
head(data1)

# Plot tree with axis in units of branch lengths and perform visual check that dates are correct

library(ape)
plot(sim.tree1)
axisPhylo()

# Load a dataset and a tree
```

```
data(data1)
data(sim.tree1)

# Find the node.ages and add as a column to the dataset
data1$age = pair.age.in.tree(dataset=data1, tree=sim.tree1, taxacolumns=c("sp1","sp2"))
head(data1)


# Plot tree with axis in units of branch lengths and perform visual check that dates are correct
library(ape)
plot(sim.tree1)
axisPhylo()
```

simulated.datasets *Simulated Datasets*

Description

Simulated datasets are provided to help users understand functions. Lineage-pairs (n=190) were created from a simulated phylogenetic tree with 20 tips. A predictor variable was generated from random draws of a standard normal. Reponse variables of various structures were simulated as follows:

1. **Simulated Dataset 1** Unbounded response, linear relationship between response and predictor, no covariance in residuals
2. **Simulated Dataset 2** Unbounded response, no relationship between response and predictor, no covariance in residuals
3. **Simulated Dataset 3** Unbounded response, linear relationship between response and predictor, covariance in residuals
4. **Simulated Dataset 4** Unbounded response, no relationship between response and predictor, covariance in residuals
5. **Simulated Dataset 5** Bounded response, linear relationship between response and predictor on link scale, no covariance in residuals
6. **Simulated Dataset 6** Bounded response, no relationship between response and predictor on link scale, no covariance in residuals
7. **Simulated Dataset 7** Bounded response, linear relationship between response and predictor on link scale, covariance in residuals
8. **Simulated Dataset 8** Bounded response, no relationship between response and predictor on link scale, covariance in residuals
9. **Simulated Lineage-Pair Covariance Matrix** A 190*190 covariance matrix used in simulating the example datasets in this package.
10. **Simulated Phylogenetic Tree** A 20-species ultrametric phylogenetic tree used in simulating the example datasets in this package.

Format

The datasets are as follows

- **data1** A data frame with 190 rows and 4 columns.
  - **sp1** Identity of Species 1 in Pair
  - **sp2** Identity of Species 2 in Pair
  - **pred** Numeric predictor variable
  - **y** Numeric response variable
- **data2** A data frame with 190 rows and 4 columns.
  - **sp1** Identity of Species 1 in Pair
  - **sp2** Identity of Species 2 in Pair
  - **pred** Numeric predictor variable
  - **y** Numeric response variable
- **data3** A data frame with 190 rows and 4 columns.
  - **sp1** Identity of Species 1 in Pair
  - **sp2** Identity of Species 2 in Pair
  - **pred** Numeric predictor variable
  - **y** Numeric response variable
- **data4** A data frame with 190 rows and 4 columns.
  - **sp1** Identity of Species 1 in Pair
  - **sp2** Identity of Species 2 in Pair
  - **pred** Numeric predictor variable
  - **y** Numeric response variable
- **data5** A data frame with 190 rows and 4 columns.
  - **sp1** Identity of Species 1 in Pair
  - **sp2** Identity of Species 2 in Pair
  - **pred** Numeric predictor variable
  - **y** Numeric response variable
- **data6** A data frame with 190 rows and 4 columns.
  - **sp1** Identity of Species 1 in Pair
  - **sp2** Identity of Species 2 in Pair
  - **pred** Numeric predictor variable
  - **y** Numeric response variable
- **data7** A data frame with 190 rows and 4 columns.
  - **sp1** Identity of Species 1 in Pair
  - **sp2** Identity of Species 2 in Pair
  - **pred** Numeric predictor variable
  - **y** Numeric response variable
- **data8** A data frame with 190 rows and 4 columns.
  - **sp1** Identity of Species 1 in Pair

- **sp2** Identity of Species 2 in Pair
- **pred** Numeric predictor variable
- **y** Numeric response variable

sim.cov.pairs A 190x190 covariance matrix

sim.tree1 A phylo object

Source

Simulated data generated with the script provided in the 'inst' directory.

`taxapair.vcv` *taxapair.vcv*

Description

Calculates an unscaled lineage-pair covariance matrix for use in downstream analyses.

Usage

```
taxapair.vcv(sp.pairs, tree, spnames=FALSE, dec=2, model="sq.diff",
  regularize=FALSE, regparam=NULL)
```

Arguments

`sp.pairs`	A table (matrix or data.frame) in which the first column contains the names of 'species 1' and the second column contains the names of 'species 2'. Names must be in the same format used in the phylogenetic tree.
`tree`	An ultrametric phylogenetic tree ('phylo' object) containing species that appear in the dataset (as either a species 1 or species 2 or both).
`spnames`	Logical determining whether to use species names as row and column labels for matrix. If TRUE, then the name of a row or column will be in the form "species1_species2". If FALSE, names are formed from the indices of the species names in the tree "indexOfSpecies1_indexOfSpecies2". Defaults to FALSE.
`dec`	Number of decimal places to round the values in the matrix; defaults to 2.
`model`	One of 'sq.diff', 'sq.sum', 'simple.sum', 'product', or 'abs.diff'. Defaults to 'sq.diff'. See details.
`regularize`	Logical indicating whether regularization should be used if resulting matrix is numerically singular; defaults to FALSE. If TRUE, regularization is conducted by adding a small value to the diagonal of the matrix. By default, this value is equal to 1% of the median value on the diagonal, which is continually added until matrix is no longer numerically singular.
`regparam`	Custom regularization parameter. Instead of adding the default 1% of the median diagonal value to the diagonal, it will add regparam * that median value. IMPORTANT: it only does this once and does not continue adding the number until the matrix is non-singular. For 2%, write 0.02; for 10%, write 0.10, and so on.

Details

Just as the traits of different species are not independent due to varying amounts of shared evolutionary history, so too are pairwise-defined traits like 'strength of RI' and 'range overlap' not independent among related taxonomic pairs. The function `taxapair.vcv()` calculates the expected covariance structure for lineage-pair datasets given the taxa in each pair and a phylogenetic tree (containing the taxa that appear in the dataset). The exact structure of this covariance depends on the underlying model by which phylogenetic signal among taxa is expected to translate into non-independence among taxonomic pairs (Anderson et al. *in review*).

Let X be some underlying continuous biological character (or set of characters) that is defined for each taxon and that has phylogenetic signal. Users can choose one of four models by which signal in X generates covariance among pairs in a lineage-pair trait:

1. sq.diff (default) – the lineage-pair trait is influenced by the square of the difference between the two taxa in X.
2. sq.sum – the lineage-pair trait is linearly by the squared sum of the value of X in each taxon.
3. simple.sum – the lineage-pair trait is influenced by the sum of the the value of X in each taxon. IMPORTANT: this model tends to result in a singular matrix.
4. product – the lineage-pair trait is influenced by the product of the the value of X in each taxon.
5. abs.diff – the lineage-pair trait is influenced by the absolute difference difference between the two taxa in X.

In each case, it is assumed that there is a linear relationship between the value calculated in each model and the response variable.

Note that simple.sum and abs.diff typically result in invalid covariance matrices, so use caution.

Value

A lineage-pair covariance matrix.

References

Anderson, S. A. S., et al. *In review*. The comparative analysis of lineage-pair data.

Examples

```
library(ape)
# Load simulated dataset and tree
data(data1)
data(sim.tree1)
# Calculate the lineage-pair covariance matrix
linpair.mat = taxapair.vcv(sp.pairs=data1[,1:2], tree=sim.tree1)
dim(linpair.mat)
# Check the validity of the matrix
covmat.check(linpair.mat)
```

`twoterm.lmm.mats` *twoterm.lmm.mats*

Description

Calculates the four matrices required for fitting the two-term lmm model of Castillo (2007). See details.

Usage

```
twoterm.lmm.mats(sp.pairs, tree)
```

Arguments

`sp.pairs`	A table (matrix or data.frame) in which the first column contains the names of 'species 1' and the second column contains the names of 'species 2'. Names must be in the same format used in the phylogenetic tree.
`tree`	An ultrametric phylogenetic tree ('phylo' object) describing the relationships among the species that appear in the dataset (as either a species 1 or species 2 or both).

Details

Castillo (2007) introduced a framework for analyzing lineage-pair data via an extension of a phylogenetic linear mixed model (plmm) in which there are two random-effect terms, one for the 'species 1' and another for the 'species 2' in every pair. The model is `Y = Xb + Z1u1 + Z1u2 + epsilon`, where b is the vector of coefficients, X is a design matrix, u1 and u2 are vectors of species-specific random effects that covary according to a phylogenetic covariance matrix C (`u1 ~ N(0,C)` and `u2 ~ N(0,C)`), Z1 and Z2 are design matrices that map the species-specific effects to the correct species in each pair, and epsilon is residual error (`epsilon~N(0,C)`). The C matrix in the model is scaled by parameters that do not come into play here.

This function takes a table containng columsn for species 1 and species 2 for every pair and a phylogenetic tree and returns the matrices Z1 and Z2 as well as pruned phylogenetic covariance matrices for u1 and u2. This pruning is sometimes required because not all species found in the dataset will appear as both 'species 1' and 'species 2'. Z1 and Z2 will therefore have different sizes and u1 and u2 require different covariance matrices. Note that the matrices themselves are pruned, not the tree from which they are derived, as the latter could result in the covariance between two species being different for the 'species 1' and 'species 2' random effects.

Value

A list of four matrices: Z1, Z2, cov1, and cov2, where the latter two describe the covariance among the random effects in u1 and u2.

References

Castillo, D. M. (2007). Factors contributing to the accumulation of reproductive isolation: A mixed model approach. Ecology and Evolution 7:5808-5820. doi.org/10.1002/ece3.3093

Examples

```
# Simulate a tree
lin.tree = phytools::pbtree(n=20)
# Generate lineage pairs as the pairwise combinations of species in the tree
lin.pairs = data.frame(t(combn(lin.tree$tip.label,2))); colnames(lin.pairs)=c("sp1", "sp2")
# Calculate the matrices
mats = twoterm.lmm.mats(sp.pairs=lin.pairs, tree=lin.tree)
# Check structure of design matrices
sapply(mats, dim)
head(mats$Z1[,1:5])
head(mats$Z2[,1:5])
head(mats$cov2[,1:5])
head(mats$cov2[,1:5])
# Ensure covariance matrices are valid covariance matrices
sapply(mats[3:4], covmat.check)
```

`twoterm.lmm.stan` *twoterm.lmm.stan*

Description

Fits the two-term lmm model of Castillo (2007) in a Bayesian framework. Bayesian sampling is conducted in the Stan software via the 'rstan' package. Users supply vectors of observations and an ultrametric phylogenetic tree. Users can alter parameters for model-parameter prior distributions and Bayesian sampling settings. See details.

Usage

```
twoterm.lmm.stan(des, y, sp1s, sp2s, tree,
  iter=2000, chains=4, coef.u=0, coef.sd=10, physig2.u=-1,
  physig2.sd=1, randsig2.loc=0, randsig2.sc=2.5, cores=4, ...)
```

Arguments

`des` A vector of predictor variable observations OR, in the case of multiple predictors, a matrix in which each column is a vector of observations of a given predictor. Function will add a column of 1s to make this a design matrix whose first column corresponds to the model intercept (unless such a column already exists).

`y` A vector of response variable observations.

`sp1s` Character vector naming the "species 1" of every pair. IMPORTANT: name formatting must match that used in the tip.label of the tree

`sp2s` Character vector naming the "species 2" of every pair. IMPORTANT: name formatting must match that used in the tip.label of the tree

`tree` Phylogenetic tree (a 'phylo' object) for the species included in the analysis (regardless of whether they are species 1 or 2).

`iter` Number of iterations to run on each chain; defaults to 2000 (more are often necessary).

`chains` Number of Markov chains to run; defaults to 4.

`coef.u` Mean of the Gaussian prior for each preditor variable coefficient; defaults to 0.

`coef.sd` SD of the Gaussian prior for each preditor variable coefficient; defaults to 10.

`physig2.u` Mean of the prior distribution (lognormal) for the scale of the phylogenetic component of residual covariance; defaults to -1.

`physig2.sd` SD of the prior distribution (lognormal) for the scale of the phylogenetic component of residual covariance; defaults to 1.

`randsig2.loc` Location parameter of the prior distribution (cauchy) for the scale of the independent component of residual covariance ; defaults to 0.

`randsig2.sc` Scale parameter of the prior distribution (cauchy) for the scale of the independent component of residual covariance ; defaults to 2.5.

`cores` Number of cores to use.

`...` additional arguments passed to `rstan::sampling`, including control parmeters (see rstan::sampling documentation)

Details

The model introduced by Castillo (2007) for analyzing lineage-pair data is a version of a phylogenetic linear mixed model (plmm) in which there are two random-effect terms, one for the ’species 1’ and another for the ’species 2’ in every pair. The model is `Y = Xb + Z1u1 + Z1u2 + epsilon`, where `u1` and `u2` are vectors of species-specific random effects that covary according to a phylogenetic covariance matrix C. In this model, `u1 ~ N(0,physig2*C)`, `u2 ~ N(0,physig2*C)`, and `epsilon~N(0, randsig2*I)`, where randsig2 is the scaling parameter for identity matrix I and physig2 is the scaling parameter for phylogenetic covariance matrix C. See `twoterm_lmm_mats` for details on calculating the Z1 and Z2 matrices.

Prior Distributions for Model Parameters: The underlying stan models assume the following prior distributions for model parameters

1. Regression coefficients: Gaussian prior (users can set prior mean and sd)
2. `physig2`: lognormal prior (users can set prior mean and sd)
3. `randsig2`: cauchy prior (users can set location and scale parameters of prior)

Value

A list containing two elements: (1) the posterior distribution of model parameters, and (2) the log-likelihood of the posteriors for use in downstream analyses (e.g. the calculation of model fitting metrics like loo or waic). For interpreting model parameters, note that `Coef[1]` is the intercept and `Coef[2]`, `Coef[3]`, ... , `Coef[N]` are regression coefficient for the 1st-Nth predictor variables.

References

Castillo, D. M. (2007). Factors contributing to the accumulation of reproductive isolation: A mixed model approach. Ecology and Evolution 7:5808-5820. doi.org/10.1002/ece3.3093

Examples

```
# Load a dataset simulated with a non-independent response observations
data(data3)
# Also load the simulated tree that was used to generate those pairs
data(sim.tree1)
# Fit an OLS model
result1 = linreg.stan(des=data3[,3], y=data3[,4], cores=2)
# Fit the twoterm.lmm.stan model
result2 = twoterm.lmm.stan(des=data3[,3], y=data3[,4], sp1s=data3[,1],
  sp2s=data3[,2], tree=sim.tree1, cores=2)
# Compare posterior parameter estimates
result1[[1]]
result2[[1]]
# Compare the fit of the two models via loo and waic
loo1 = loo::loo(result1[[2]])
loo2 = loo::loo(result2[[2]])
waic1 = loo::waic(result1[[2]])
waic2 = loo::waic(result2[[2]])
loo1
loo2
waic1
waic2
loo::loo_compare(loo1, loo2)
loo::loo_compare(waic1, waic2)
```

Index