Package 'caper'

April 17, 2018

Type Package

Title Comparative Analyses of Phylogenetics and Evolution in R

Version 1.0.1

Date 2018-04-16

Author David Orme, Rob Freckleton, Gavin Thomas, Thomas Petzoldt, Susanne Fritz, Nick Isaac, Will Pearse

Maintainer David Orme <d.orme@imperial.ac.uk>

Imports stats, graphics, utils, methods

Depends R (>= 2.10), ape (>= 3.0-6) , MASS, mvtnorm

Suggests xtable

Description Functions for performing phylogenetic comparative analyses.

License GPL (>= 2)

Repository CRAN

Repository/R-Forge/Project caper

Repository/R-Forge/Revision 124

Repository/R-Forge/DateTimeStamp 2018-04-16 16:09:33

Date/Publication 2018-04-17 10:18:36 UTC

NeedsCompilation no

R topics documented:

`caper-package` *Comparative analysis of phylogenetics and evolution in R (caper)*

Description

The **caper** package provides a set of functions to conduct comparative analyses using both independent contrasts and phylogenetic generalised least squares methods. It also provides functions to combine phylogeny objects with data frames into simple comparative datasets and some utility functions for manipulating phylogenies.

In addition to linear models correcting for phylogeny, the package also provides functions to test tree imbalance, calculate various measures of phylogenetic diversity and simulate phylogenies and traits.

Details

Package:	caper
Type:	Package
Version:	0.2
Date:	2011-05-06
License:	GPL

A package to carry out phylogenetic comparative analysis.

Author(s)

David Orme Maintainer: David Orme <d.orme@imperial.ac.uk>

See Also

ape

`anova.caic` *Anova and model checking methods for independent contrast models.*

Description

These functions provide ANOVA tables and model comparison using ANOVA and AIC, along with standard model diagnostic plots and accessor functions for phylogenetic independent contrast objects.

Usage

```
## S3 method for class 'caic'
anova(object, ...)
## S3 method for class 'caiclist'
anova(object, ..., scale=0, test='F')
## S3 method for class 'caic'
logLik(object, ...)
## S3 method for class 'caic'
predict(object, ...)
## S3 method for class 'caic'
residuals(object, ...)
## S3 method for class 'caic'
coef(object, ...)
## S3 method for class 'caic'
plot(x, ...)
```

Arguments

`object`	An object of class 'caic'.
`scale`	A character string specifying the test statistic to be used. Can be one of "F", "Chisq" or "Cp", with partial matching allowed, or NULL for no test.
`test`	numeric. An estimate of the noise variance sigma^2. If zero this will be estimated from the largest model considered.
`x`	An object of class 'caic'.
`...`	Further argument to be passed to methods.

Details

The 'anova' method provides access to single anova tables for a model and to comparison of lists of models. The 'logLik' method provides access to the log likelihood of the 'caic' model and hence to AIC comparison of models.

The 'plot' method uses the standard set of model diagnostic plots for linear models. It is also wise to check the evolutionary assumptions of independent contrast models using the 'caic' specific diagnostic plots. The 'predict' and 'residuals' functions provide access to these parts of the 'caic' object.

Author(s)

David Orme

See Also

crunch, brunch,macrocaic,caic.diagnostics

Examples

```
data(shorebird)
shorebird.data$lgEgg.Mass <- log(shorebird.data$Egg.Mass)
shorebird.data$lgM.Mass <- log(shorebird.data$M.Mass)
shorebird.data$lgF.Mass <- log(shorebird.data$F.Mass)
shorebird <- comparative.data(shorebird.tree, shorebird.data, Species)

cMod1 <- crunch(lgEgg.Mass ~ lgM.Mass * lgF.Mass, data=shorebird)
cMod2 <- crunch(lgEgg.Mass ~ lgM.Mass + lgF.Mass, data=shorebird)
cMod3 <- crunch(lgEgg.Mass ~ lgM.Mass , data=shorebird)

anova(cMod1, cMod2, cMod3)
AIC(cMod1, cMod2, cMod3)

plot(cMod3)
```

---

anova.pgls *Anova and AIC tables for 'pgls' models.*

Description

The 'anova' function creates ANOVA tables for a 'pgls' models using sequential sums of squares.

Usage

```
## S3 method for class 'pgls'
anova(object, ...)
## S3 method for class 'pglslist'
anova(object, ..., scale = 0, test = "F")
```

Arguments

object	A 'pgls' model object.
...	Additional 'pgls' models.
scale	A character string specifying the test statistic to be used. Can be one of "F", "Chisq" or "Cp", with partial matching allowed, or NULL for no test.
test	numeric. An estimate of the noise variance sigma^2. If zero this will be estimated from the largest model considered.

Details

The sequential sums of squares are calculated by refitting the model in the order of the terms of the formula and so can take a little time to calculate. Branch length transformations are held at the values of the initial object. The 'logLik.pgls' provides a simple accessor function that allows the use of AIC model comparisons. Note that the generic AIC methods do no checking to ensure that sensible models are being compared.

Value

A table of class 'anova' and 'data.frame' that employs the generic plot methods for 'anova' tables.

Note

The functions build heavily on the generic methods 'anova.lm' and 'anova.lmlist'.

Author(s)

Rob Freckleton, David Orme

See Also

pgls

Examples

```
data(shorebird)
shorebird <- comparative.data(shorebird.tree, shorebird.data, Species, vcv=TRUE, vcv.dim=3)

mod1 <- pgls(log(Egg.Mass) ~ log(M.Mass) * log(F.Mass), shorebird)
anova(mod1)

mod2 <- pgls(log(Egg.Mass) ~ log(M.Mass) + log(F.Mass), shorebird)
mod3 <- pgls(log(Egg.Mass) ~ log(M.Mass) , shorebird)
mod4 <- pgls(log(Egg.Mass) ~ 1, shorebird)

anova(mod1, mod2, mod3, mod4)
AIC(mod1, mod2, mod3, mod4)
```

BritishBirds *Conservation status of British birds (Thomas 2008)*

Description

The dataset contains a molecular phylogeny of 249 species of British birds and a data frame containing information on the conservation status of 181 of those species. The dataset is taken from Thomas (2008)

Usage

```
data(BritishBirds)
```

Details

The data frame contains 26 variables:

binomial common_name Red_list Amber_list Green_list Red_amber_list IUCN Red_HD Amber_HD Red_amber_HD Red_list_BDp amb_BDp Red_amb_BDp Red_list_BDr Red_list_WDp amb_BDr amb_WDMp amb_spec amb_BR amb_BL amb_WL amb_BI amb_WI Red_amb_BDr pop_size range_size

References

Thomas, G. H. (2008). Phylogenetic distributions of british birds of conservation concern. Proceedings of the Royal Society B-Biological Sciences, 275(1647):2077-2083.

Examples

```
data(BritishBirds)
BritishBirds <- comparative.data(BritishBirds.tree, BritishBirds.data, binomial)
```

brunch *Comparative analysis using the brunch algorithm.*

Description

Calculate a linear model using the brunch algorithm.

Usage

```
brunch(formula, data, phy, names.col, stand.contr = TRUE, robust = Inf,
       ref.var=NULL, node.depth=NULL, equal.branch.length=FALSE)
```

Arguments

formula	A model formula.
data	An 'comparative.data' object. Alternatively, a data frame.
phy	An object of class 'phylo', required when data is not a 'comparative.data' object.
names.col	A name specifying the column in 'data' that matches rows to tips in 'phy', required when data is not a 'comparative.data' object.
stand.contr	A logical flag indicating whether or not to standardize contrasts.
robust	A threshold value of studentized residuals to exclude from the model.
ref.var	A reference variable present in the model that is used to specify the direction of calculation of contrasts. If null, this is assumed to be the first explanatory variable.
node.depth	A positive integer greater than 1 used to restrict the model to contrasts with a node depth less than or equal to the specified depth. Tips have a depth of 1.
equal.branch.length	If set to 'TRUE' then all branch lengths are set to 2.

Details

This function implements the 'brunch' algorithm for modelling the relationship between variables that are phylogenetically non-independent. This method was described and previously implemented in the Mac Classic computer programs CAIC, written by Purvis and Rambaut (1995) and updated by Nick Isaac and Paul-Michael Agapow.

The 'brunch' algorithm calculates contrasts for models that include binary categorical variables. Contrasts are identified and calculated for all variables in the model for a set of nodes where each side can be unequivocally attributed to one or other of the categories. Unlike 'crunch', nested contrasts are not calculated and each row of data at the tips is used only once. This follows Burt (1989): contrasts whose paths do not meet or cross at any point will be phylogenetically independent.

Factors with more than two levels are supported but *must* be ordered to allow sensible contrasts to be drawn. In addition, there is no single best compromise set of contrasts with non-binary factors and implementations may differ in the set chosen.

The user provides a comparative dataset. The formula specifies the model to be fitted and contrasts are calculated in those variables. The specified reference variable is used to ensure that contrasts for multivariate models are calculated in a consistent direction at each node. The function `brunch` acts as a data preparation wrapper for the function `contrCalc`, which is not intended to be directly called by users. Missing data can be present in the explanatory variables: the algorithm makes use of the complete data available at each node as was the case with CAIC.

Polytomies - more detail here The Mac Classic program CAIC used 1 for both 'Brunch' and 'Crunch' analyses and this the default.

Value

A object of class 'caic'.

Author(s)

David Orme

References

Purvis, A. and Rambaut, A. (1995) Comparative analysis by independent contrasts (CAIC): an Apple Macintosh application for analysing comparative data. Computer Appl. Biosciences 11, 247-251.

Burt, A. (1989). Comparative methods using phylogenetically independent contrasts. Oxford Surveys in Evolutionary Biology, 6:33-53.

See Also

caic-class for 'caic' object structure and methods.

Examples

```
data(perissodactyla)
perisso <- comparative.data(perissodactyla.tree, perissodactyla.data, Binomial)
brunchMod <- brunch(log.female.wt ~ Territoriality, data=perisso)
summary(brunchMod)

# plot the contrasts
brunchTab <- caic.table(brunchMod)
plot(log.female.wt ~ Territoriality, brunchTab)

# for the actual model diagnostics
par(mfrow=c(3,1))
caic.diagnostics(brunchMod)
```

---

caic-class *The 'caic' S3 object class and methods*

Description

The functions 'crunch', 'brunch', 'macrocaic' and 'piclm' all return an object containing an independent contrast model. The structure of the object and the available methods are described here.

Format

A 'caic' object is a list containing the following:

contrast.data A list of the following:

- **contr** A list containing matrices of the contrasts in the response variable (contr\$response) and explanatory variables (contr\$explanatory).
- **nodalVals** A list containing matrices of the nodal values in the response variable (contr\$response) and explanatory variables (contr\$explanatory).
- **contrVar** A numeric vector of the expected variance for each contrast.
- **nChild** A vector showing the number of nodes descending from each internal node

nodeDepth A vector showing the maximum number of nodes between each internal node and the tips of the phylogeny (including both the node in question and the tip and hence always >=2)

validNodes A logical vector showing which internal nodes on the tree have valid contrasts, given the available data and any user constraints.

data A 'comparative.data' object containing the phylogeny used to calculate contrasts and the original data.

lm An 'lm' object containing a regression model through the origin for the calculated contrast

In addition, the object may have the following attributes:

contr.method One of 'crunch', 'brunch' or 'piclm'.

macro Either 'RRD' or 'PDI' if the response contrasts are calculated as species richness contrasts using `macrocaic`

stand.cont A logical value showing whether the contrasts in the object have been standardized.

`caic.diagnostics` *Diagnostic tools for independent contrasts models*

Description

These functions are a set of diagnostic tools to assess whether a particular contrast model is adequate.

Usage

```
caic.table(caicObj, validNodes=TRUE, nodalValues=FALSE, ultrametric.tol=0.0001,
           CAIC.codes=FALSE, style="CAIC")
caic.diagnostics(caicObj, which.terms=NULL, which.tests=c("NV","SD","AGE"),
           plot=TRUE, outlier=3, ultrametric.tol=0.0001, plot.signif=plot,
           alpha=0.05, cex.id=0.7,  ...)
## S3 method for class 'caic.diagnostics'
print(x, ...)
caic.robust(caicObj, robust=3)
caic.label(phy, charset=NULL, action="insert", style="CAIC", tips=FALSE)
```

Arguments

`caicObj`	An object of class 'caic'
`validNodes`	A logical indicating whether to return only the nodes at which valid contrasts have been drawn or all nodes in the phylogeny.
`nodalValues`	A logical indicating whether to include the estimated nodal values in the contrasts table.
`ultrametric.tol`	A value in branch length units indicating how large a departure from ultrametric path lengths is permitted before calculating node ages is blocked.

CAIC.codes	Should CAIC style node labels be added to the contrast table?
style	Either 'CAIC', which is the default, or 'RLE' for run-length encoded codes.
which.terms	A character vector of the terms in the linear model for which diagnostic checks should be made. The default is to use all terms.
which.tests	A character vector of the tests to conduct. Any of 'NV', 'SD' and 'Age' may be selected and the default is to use all three.
plot	A logical indicating whether or not to create plots of the diagnostic tests.
outlier	A value indicating the size of absolute studentized residuals above which a contrast will be highlighted in diagnostic plots
robust	A value indicating the size of absolute studentized residuals above which a contrast will be removed from the contrast model.
plot.signif	By default, significant relationships are indicated on the plots by showing the model line.
alpha	This value sets the significance at which model predictions are added to the diagnostic plots.
cex.id	Adjust the size of labels added to outlier points.
x	An object of class 'caic diagnostics' to be printed out
...	Generic arguments to print.caic.diagnostics or plot arguments for print.caic.diagnostics.
phy	A 'phylo' object.
charset	A vector of characters to be used to construct CAIC codes.
action	One of 'replace', 'append' or 'insert'.
tips	A logical indicating whether to modify the tip labels of the phylogeny in addition to the internal node labels.

Details

The most general function (caic.table) extracts a data frame from a 'caic' object that contains the contrasts, expected variance ('contrVar'), node depth, number of descendant lineages and studentized residuals. It can also optionally include the nodal values estimated at internal nodes and CAIC style node labels. This data frame is very similar to the output files generated by CAIC and MacroCAIC. By default, only the valid nodes contributing contrasts to the model are shown but other internal nodes can also be included ('validNodes=FALSE').

The caic.diagnostics function carries out a set of regression tests on the absolute values of contrasts in a model. These include regression against estimates of the nodal values ('NV), standard deviation at each node ('SD') and, where the underlying phylogeny is ultrametric, the log age at each node ('AGE'). Significant regressions indicate problems with the distribution of the contrast values. The user can select which of these plots are to be generated and also which terms in the contrasts model will be plotted. The function can also plot these relationships and show the slope of problematic tests (slope significance < alpha).

Plots from the caic.diagnostics function will plot outliers in red and label these points. Outliers are identified as points with absolute studentized residuals greater than the cutoff specified in 'outlier'. The function caic.robust is a simple method to obtain a refitted 'caic' object model from which these outlying contrasts have been excluded.

The `caic.label` function provides a method to label nodes (and optionally tips) with CAIC phylogeny codes. These are simple alphabetic sequences that indicate the branching structure of the tree from the root and are unique for each node. With deeply nested trees, these codes can be very long and so the option 'RLE' is provided to reduce the length of codes for plotting onto trees: for example, the code AABAAAC would be converted to 2AB3AC. If `action` is 'replace', then the phylogeny tip and node labels are replaced with the CAIC codes. If `action` is 'append', then the codes are appended onto the end of existing labels, with the internal node numbers used if no node labels exist. Finally, if `action` is 'insert', then a named character vector called 'edge.caic.code' is inserted into the 'phylo' object list.

Value

`caic.table` A data frame of contrasts and other nodal values.

`caic.diagnostics` An array of slope coefficients for each test (rows) and for each term (depth).

`caic.robust` A 'caic' object with a model omitting outliers.

`caic.labels` A 'phylo' object with modified labels.

Author(s)

David Orme

See Also

`crunch`, `brunch`, `macrocaic`

Examples

```
data(shorebird)
shorebird <- comparative.data(shorebird.tree, shorebird.data, Species)
crunchMod <- crunch(log(Egg.Mass) ~  log(M.Mass) + log(F.Mass), data=shorebird)
caic.diagnostics(crunchMod)
```

---

caper-benchmarks *Datasets used for benchmarking caper*

Description

These data files and model objects contain input datasets for benchmarking the caper package against other implementations and the results returned by those other implementations. For more details, see the benchmark vignette for caper.

See Also

vignette()

clade.matrix *Create a clade matrix from a phylogeny*

Description

Takes a phylogeny in the 'ape' package format and converts it into a binary matrix showing which tips (matrix columns) subtend from each node in the phylogeny (matrix rows). This is a useful format for quickly calculating branch length information for subsets of the phylogeny.

Usage

```
clade.matrix(phy)
```

Arguments

phy	A object of class 'phylo'

Details

The clade matrix shows the tips from a phylogeny that subtend from each internal and external node. Each tip is represented as column showing the nodes of which it is a member and hence each row shows the tips that are members of a given node. Dropping columns gives a quick and easy way to find out which edges are retained in a particular subset of the tree and this structure is used for quickly calculating branch lengths calculations or clade statistics.

Value

A list of class 'clade.matrix' containing the following components:

clade.matrix	A binary m x n matrix, where m is the total number of nodes in the phylogeny and n is the number of tips. An element is 1 if tip n_i subtends from a node m_j.
edge.length	A numeric vector of length m showing the edge length leading to each node in the phylogeny and named with the node number.
tip.label	A character vector of length n giving the labels assigned to the tips of the phylogeny.
edge	The edge matrix from the original phylogeny.

Author(s)

David Orme

Examples

```
data(perissodactyla)
clade.matrix(perissodactyla.tree)
```

clade.members *Identify tips descended from a node*

Description

Obtains a vector of the tips subtending from either one node or all nodes in a phylogeny.

Usage

```
clade.members(x, phy, tip.labels = FALSE, include.nodes=FALSE)
clade.members.list(phy, tips = FALSE, tip.labels = FALSE, include.nodes=FALSE)
```

Arguments

x	A integer identifying the node for which a list of tips is required.
phy	An object of class 'phylo'.
tips	A logical indicating whether to include external node membership in the list.
tip.labels	A logical flag indicating whether to return the node numbers of the tips or their tip labels.
include.nodes	A logical flag indicating whether to return the node number of descendent internal nodes

Details

The function clade.members.list runs clade.members over each node in the phylogeny, possibly including the external nodes as indicated by the tips argument, and returns a list of vectors showing the members of the clade defined by each node.

Value

A numeric vector of external node (i.e. tip) numbers or a character vector of tip labels for a single internal node or, for clade.members.list, a list of such vector for all nodes in the phylogeny. If include.nodes is TRUE then clade.members returns a list of length two containing a vector of the descendent tips and a vector of the descendent internal nodes - clade.members.list then contains a list of such lists.

Author(s)

David Orme, Lynsey McInnes

Examples

```
data(perissodactyla)
# use comparative.data to add node labels
perisso <- comparative.data(perissodactyla.tree, perissodactyla.data, Binomial, na.omit=FALSE)
plot(perisso$phy, show.node.label=TRUE)
clade.members(22, perisso$phy, tip.labels=TRUE)
```

```
clade.members.list(perisso$phy, tip.labels=FALSE)
```

`comparative.data` *Comparative dataset creation*

Description

A simple tool to combine phylogenies with datasets and ensure consistent structure and ordering for use in functions.

Usage

```
comparative.data(phy, data, names.col, vcv=FALSE, vcv.dim=2, na.omit=TRUE,
             force.root=FALSE, warn.dropped=FALSE, scope=NULL)
## S3 method for class 'comparative.data'
print(x, ...)
## S3 method for class 'comparative.data'
na.omit(object, scope=NULL, ...)
## S3 method for class 'comparative.data'
subset(x, subset, select, ...)
## S3 method for class 'comparative.data'
reorder(x, order, ...)
## S3 method for class 'comparative.data'
x[i, j]
as.comparative.data(x, ...)
caicStyleArgs(phy, data, names.col, warn.dropped=FALSE)
```

Arguments

`data`	A data frame containing variables that can be attributed to the taxa at the tips of a phylogeny.
`phy`	A phylogeny (class 'phylo') to be matched to the data above.
`names.col`	The name of a column in the provided data frame that will be used to match data rows to phylogeny tips.
`vcv`	A logical value indicating whether to include a variance covariance array representing the phylogeny within the comparative dataset.
`vcv.dim`	Either 2 (a standard VCV matrix) or 3 (an array retaining the individual branches contributing to the standard values). The array form is of use for optimising some branch length transformations.
`na.omit`	A logical value indicating whether to reduce the comparative dataset to those tips for which all selected variables are complete. Note that some functions cannot handle missing data and will return an error.
`force.root`	Many functions consider a basal polytomy to indicate an unrooted tree. Using force.root=TRUE will set an arbitrary root edge below this polytomy.

warn.dropped	A logical value indicating whether to warn the user when data or tips are dropped in creating the comparative data object.
scope	A model formula, used to indicate which variables to consider when omitting row containing NA values.
x	An object of class 'comparative.data'.
object	An object of class 'comparative.data'.
subset	A logical expression indicating rows of data to keep: missing values are taken as false.
select	An expression, indicating columns to select from the data frame.
order	One of 'cladewise' or 'pruningwise'. See reorder.phylo.
i,j	Indices specifying tips or data columns to extract. See details.
...	Further arguments to functions.

Details

The function matches rows in a data frame to tips on a phylogeny and ensures correct ordering of the data with respect to the tips. It also can add a variance covariance representation of the phylogeny. Mismatched rows and tips are removed and the taxon labels of these are stored in the 'dropped' slot of the 'comparative.data' object. The 'print' method displays a brief summary of the dataset contents and the names of the original 'phylo' and 'data.frame' objects. If any rows or tips were dropped, 'print' will also show a venn diagram of the data shared and dropped from each source. Node labels are preserved but must be unique - unlabelled nodes will be assigned numeric codes.

The 'na.omit' and 'subset' methods provide simple ways to clean up and extract parts of the comparative dataset. In particular, 'subset' acts exclusively with the data component of the object and, like subset on a data frame, expects the subset argument to produce a logical vector of data rows to include. The 'reorder' method is use to restructure all the components with the 'comparative.data' object into either pruningwise or cladewise order. This uses code from the 'ape' library: see reorder.phylo.

The '[' method allows subsets to be taken of the data. There are no replace methods ('[<-'). If only one index is specified (e.g. x[2]), then this is interpreted as extracting data columns from the object. Otherwise (e.g. x[2,], x[1,1]), the first index will specify tips to extract and the second index will specify columns. Indices for tips are permitted to be numeric, logical or character vectors or empty (missing) or NULL. Numeric values are coerced to integer as by as.integer (and hence truncated towards zero). Character vectors will be matched to the names of the object (or for matrices/arrays, the dimnames): see 'Character indices' below for further details.

The function 'caicStyleArgs' handles turning 'phy', 'data' and 'names.col' arguments into a 'comparative.data' object when they are provided separately to a function. This argument structure was used in older versions of many functions.

All of these functions are in part a substitute for the considerably more sophisticated handling of such data in the package 'phylobase', which will be integrated into later releases.

Value

A list of class 'comparative.data':

phy	An object of class 'phylo'

data	A data frame of matched data
data.name	The original object name of the data
phy.name	The original object name of the phylogeny
dropped	A list of taxon names dropped from the dataset:
	unmatched.rows Data rows that do not match to tips
	tips Tips that do not match to data rows

And optionally:

vcv	A variance covariance matrix of the phylogeny
vcv.dim	The dimension of the VCV - 2 for a standard VCV matrix and 3 for an expanded array retaining individual branch lengths

Author(s)

David Orme

See Also

crunch,pgls

Examples

```
data(shorebird)
shorebird <- comparative.data(shorebird.tree, shorebird.data, 'Species')
print(shorebird)

subset(shorebird, subset=Mat.syst == 'MO')

sandpipers <- grep('Calidris', shorebird$phy$tip.label)
shorebird[-sandpipers, ]

sandpipers <- grep('Calidris', shorebird$phy$tip.label, value=TRUE)
shorebird[sandpipers, ]

shorebird[]
shorebird[,]
shorebird[2:3]
shorebird[, 2:3]
shorebird[1:15, ]
shorebird[1:15, 2:3]
```

crunch *Comparative analysis using the crunch algorithm.*

Description

Calculate a linear model using the crunch algorithm.

Usage

```
crunch(formula, data, phy, names.col, stand.contr = TRUE, robust=Inf,
   ref.var=NULL, node.depth=NULL, polytomy.brlen=0, equal.branch.length=FALSE,
   factor.action="abort")
```

Arguments

formula	A model formula.
data	An 'comparative.data' object. Alternatively, a data frame.
phy	An object of class 'phylo', required when data is not a 'comparative.data' object.
names.col	A name specifying the column in 'data' that matches rows to tips in 'phy', required when data is not a 'comparative.data' object.
stand.contr	A logical flag indicating whether or not to standardize contrasts.
robust	A threshold value of studentized residuals to exclude from the model.
ref.var	A reference variable present in the model that is used to specify the direction of calculation of contrasts. If null, this is assumed to be the first explanatory variable.
node.depth	A positive integer greater than 1 used to restrict the model to contrasts with a node depth less than or equal to the specified depth. Tips have a depth of 1.
polytomy.brlen	The internal branch length used for calculating contrasts at a polytomy, following Pagel's (1992) method.
equal.branch.length	If set to 'TRUE' then all branch lengths are set to 2.
factor.action	One of "abort", "warn" or "allow", describing whether to stop if the formula contains a factor ("abort"), or continue after converting the factor to a numeric variable, either with ("warn") or without ("allow") a warning.

Details

This function implements the 'crunch' algorithm for modelling the relationship between variables that are phylogenetically non-independent. The method was first described by Felsenstein (1985) and subsequently extended to permit the use of phylogenies with polytomies by Pagel (1992). This method was previously implemented in the Mac Classic computer programs CAIC, written by Andy Purvis, Andy Rambaut (Purvis and Rambaut, 1995) and updated by Nick Isaac and Paul-Michael Agapow.

The user provides a comparative dataset. The formula specifies the model to be fitted and contrasts are calculated in those variables. The specified reference variable is used to ensure that contrasts for multivariate models are calculated in a consistent direction at each node. The function `crunch()` acts as a data preparation wrapper for the function `contrCalc()`, which is not intended to be directly called by users. Missing data can be present in the explanatory variables: the algorithm makes use of the complete data available at each node as was the case with CAIC.

The resulting table of contrasts is then used to fit the specified model - note that the intercept is automatically dropped from the model if present, following REF HERE.

Contrasts at polytomies are calculated following Pagel (1992). The descendants from the node are split into two groups based on whether they are above or below the group mean in the reference variable. If there is no variation in the reference variable, then a 1:(N-1) split is used. Weighted means in the variables are then calculated for each subgroup and a contrast is calculated between these values using an arbitrary internal branch length.

Value

A object of class 'caic'.

Warning

At a polytomy, subtracting the internal branch length from the real branch lengths can lead to negative branch lengths. CAIC used a hard-coded internal branch length of 1 for calculating crunch contrasts at polytomies. From version 2.6.9, CAIC issued a warning if this lead to negative branch lengths but allowed contrast calculation to continue. In contrast, the implementation in `crunch()` uses a default internal branch length (`polytomy.brlen`) of 0 and also treats a negative branch length in a polytomy calculation as an error. In either case, contrast calculation on negative branch lengths is not a desirable outcome. Duplication of CAIC results therefore requires `polytomy.brlen` to be set to 1 and an analyis *cannot* be duplicated precisely if the phylogeny contains polytomies with descending branches shorter than 1. The method used by `pic.lm` to handle polytomies avoids such problems.

Author(s)

David Orme

References

Felsenstein, J. (1985). Phylogenies and the comparative method. Am. Nat. 125, 1-15

Pagel, M. D. (1992). A method for the analysis of comparative data. J. theor. Biol. 156, 431-442.

Purvis, A. and Rambaut, A. (1995) Comparative analysis by independent contrasts (CAIC): an Apple Macintosh application for analysing comparative data. Computer Appl. Biosciences 11, 247-251.

See Also

`caic-class` for 'caic' object structure and methods.

Examples

```
data(shorebird)
shorebird <- comparative.data(shorebird.tree, shorebird.data, Species)
crunchMod <- crunch(Egg.Mass ~ F.Mass + M.Mass, data=shorebird)
summary(crunchMod)
# plot the contrasts
crunchTab <- caic.table(crunchMod)
plot(Egg.Mass ~ F.Mass, crunchTab)
# for the actual model diagnostics
par(mfrow=c(3,2))
caic.diagnostics(crunchMod)
```

`fusco.test` *Imbalance statistics using Fusco and Cronk's method.*

Description

Fusco and Cronk (1995) described a method for testing the imbalance of phylogenetic trees based on looking at the distribution of I. I is calculated using the number of tips descending from each side of a bifurcating node using the formula I = (B-m)/(M-m) and is bounded between 0 (a perfectly balanced node) and 1 (maximum imbalance). B is the larger number of tips descending from each branch, M is the maximum size of this larger group (i.e. a 1 : (S-1) split, where S is the total number of descendent tips), and m is the minimum size of the larger group (ceiling of S/2). The method can cope with small proportions of polytomies in the phylogeny and these are not used in calculating balance statistics. It can also incorporate information about species richness at the tips of the phylogeny and can therefore be used to distinguish between an unbalanced topology and the unbalanced distribution of diversity at the tips of a phylogeny.

Purvis et al. (2002) demonstrated that I is not independent of the node size S, resulting in a bias to the expected median of 0.5. They proposed a modification (I') that corrects this to give a statistic with an expected median of 0.5 regardless of node size. The defaults in this function perform testing of imbalance using I', but it is also possible to use the original measure proposed by Fusco and Cronk (1995).

Usage

```
fusco.test(phy, data , names.col, rich, tipsAsSpecies=FALSE,
       randomise.Iprime=TRUE,  reps=1000, conf.int=0.95)
## S3 method for class 'fusco'
print(x, ...)
## S3 method for class 'fusco'
summary(object, ...)
## S3 method for class 'fusco'
plot(x, correction=TRUE, nBins=10, right=FALSE, I.prime=TRUE, plot=TRUE, ...)
```

Arguments

phy	An object of class 'comparative.data' or of class 'phylo'.
data	A data frame containing species richness values. Not required if phy is a 'comparative.data' object.
names.col	A variable in data identifying tip labels. Not required if phy is a 'comparative.data' object.
rich	A variable identifying species richness.
tipsAsSpecies	A logical value. If TRUE, a species richness column need not be specified and the tips will be treated as species.
randomise.Iprime	Use a randomization test on calculated I' to generate confidence intervals.
reps	Number of replicates to use in simulation or randomization.
conf.int	Width of confidence intervals required.
x, object	An object of class 'fusco'.
correction	Apply the correction described in Appendix A of Fusco and Cronk (1995) to the histogram of nodal imbalance.
nBins	The number of bins to be used in the histogram of nodal imbalance.
right	Use right or left open intervals in plotting the distribution and calculating the correction
I.prime	Plot distribution of I' or I.
plot	If changed to FALSE, then the plot method does not plot the frequency histogram. Because the method invisibly returns a table of histogram bins along with the observed and corrected frequencies, this isn't as dim an option as it sounds.
...	Further arguments to generic methods

Details

I is calculated only at bifurcating nodes giving rise to more than 3 tips (or more than 3 species at the tips): nodes with three or fewer descendants have no variation in I and are not informative in assessing imbalance. The expected distribution of the nodal imbalance values between 0 and 1 is theoretically uniform under a Markov null model. However, the range of possible I values at a node is constrained by the number of descendent species. For example, for a node with 8 species, only the values 0, 0.33, 0.66, 1 are possible, corresponding to 4:4, 5:3, 6:2 and 7:1 splits (Fusco and Cronk, 1995). As node size increases, this departure from a uniform distribution decreases. The plot method incorporates a correction, described by Fusco and Cronk (1995), that uses the distribution of all possible splits at each node to characterize and correct for the departure from uniformity.

The randomization option generates confidence intervals around the mean I'.

Value

The function fusco.test produces an object of class 'fusco' containing:

observed	A data frame of informative nodes showing nodal imbalance statistics. If the phylogeny has labelled nodes, then the node names are also returned in this data frame.
median	The median value of I.
qd	The quartile deviation of I.
tipsAsSpecies	A logical indicating whether the tips of the trees were treated as species or higher taxa.
nInformative	The number of informative nodes.
nSpecies	The number of species distributed across the tips.
nTips	The number of tips.
reps	The number of replicates used in randomization.
conf.int	The confidence levels used in randomization.

If randomise.Iprime is TRUE, or the user calls fusco.randomize on a 'fusco' object, then the following are also present.

randomised	A data frame of mean I' from the randomized observed values.
rand.mean	A vector of length 2 giving confidence intervals in mean I'.

Author(s)

David Orme, Andy Purvis

References

Fusco, G. & Cronk, Q.C.B. (1995) A New Method for Evaluating the Shape of Large Phylogenies. J. theor. Biol. 175, 235-243

Purvis A., Katzourakis A. & Agapow, P-M (2002) Evaluating Phylogenetic Tree Shape: Two Modifications to Fusco & Cronk's Method. J. theor. Biol. 214, 93-103.

Examples

```
data(syrphidae)
syrphidae <- comparative.data(phy=syrphidaeTree, dat=syrphidaeRich, names.col=genus)
summary(fusco.test(syrphidae, rich=nSpp))
summary(fusco.test(syrphidae, tipsAsSpecies=TRUE))
plot(fusco.test(syrphidae, rich=nSpp))
```

fuscoData *Example dataset for Fusco imbalance calculations*

Description

This dataset contains the phylogeny of bird families and species richness originally included with the FUSCO imbalance calculation programs.

Usage

```
data(fuscoData)
```

Format

The dataset provides a phylogeny (`fuscoBirdTree`) and a data frame (`fuscoBirdData`). The phylogeny is a 137 tip tree containing one polytomy and the dataframe provides tip labels and species richness for each bird family. This dataset was provided with the original DOS implementation of the test and the families were unlabelled in this original file.

See Also

fusco.test

growTree *Tree simulation with traits.*

Description

This function provides a very general environment in which to simulate trees. The basic philosophy is that the user provides a series of expressions that define speciation rates, extinction rates and trait evolution. These expressions can make use of information about the internal state of the tree, allowing for very flexible definitions of rules for tree growth.

Usage

```
growTree(b = 1, d = 0, halt = 20, grain = 0.1, linObj = NULL, ct.start = NULL,
         ct.change = NULL, ct.var = NULL, dt.rates = NULL, inheritance = NULL,
         trace.events = FALSE, trace.cladesize = FALSE, output.lineages = FALSE,
         neg.rates = "abort", inf.rates = "abort", stall.time = 10,
         extend.proportion=0)
## S3 method for class 'growTree'
as.comparative.data(x, ...)
```

Arguments

b	A speciation rate. This can be a numeric constant, as in the default, which specifies a single speciation rate for the simulation. Alternatively, this can be an expression, or a list of expressions which define speciation rate in terms of the properties of the tree. See details for discussion of those properties.
d	An extinction rate, described as above.
halt	A rule use to halt the simulation. The default is the number of tips in the simulation, specified as a single integer, but this can also be an expression or list of expressions on the properties of the tree. The simulation is halted when any of these expressions becomes true.
grain	Where rates depend on time or trait values, it becomes necessary to allow time to pass discretely in order to re-evaluate waiting times under the changing values. This sets the amount of time that is allowed to pass before re-evaluation. If rates do not depend on such changing parameters, it is sensible to set this to infinity - this will ensure that the flow of the simulation is not slowed by checking.
linObj	This can be used to supply an existing simulation object, which will then continue to grow under the provided rules. This allows the user to simulate trees with different sets of rules operating in different epochs. The function `linToApe` will convert such an object to a 'phylo' object, retaining additional trait data as extra components of the 'phylo' object list.
ct.start	A numeric vector specifying the starting values for continuous traits. If unnamed these will be sequentially named as 'ct1', 'ct2' etc. The names of traits may be used in expressions governing tree growth rules.
ct.change	A numeric vector describing the mean change per unit time in continuous trait values, used to simulate a directional bias in character evolution. If ct.change is NULL, then this is assumed to be zero for each species.
ct.var	Either a vector of variances for each trait or a square matrix describing the variances and covariances amongst the continuous traits. If this is NULL, then uncorrelated traits with a variance of 1 are assumed.
dt.rates	A list of matrices describing the rate of transition between discrete character traits. Each matrix defines a trait and, as with ct.start, the list names are used to identify the traits in the simulation and default to 'dt1', 'dt2', etc. The dimnames of the matrix are used to identify the states of the trait and default to 'st1', 'st2', etc. The matrix need not be symmetrical: the rates are defined from the states in the columns to the states in the rows, hence the diagonal should probably be zero. Each trait is assumed to start the simulation in the first state in the matrix.
inheritance	A list of rules that are applied after a speciation and can be used to modify trait values for the descendent lineages. The names of the list specify which traits are to be modified and, for each trait specified, should return a vector of length two which replaces the existing values.
trace.events	A logical flag, indicating whether or not report speciation, extinction and discrete character evolution events.
trace.cladesize	A positive integer giving an increment size for the simulation to report clade size if required.

`output.lineages`
: A logical flag indicating whether to return the internal lineages object.

`neg.rates`
: One of 'abort', 'warn' or 'quiet', defining the behaviour when a rate calculation produces a negative number. With 'warn' and 'quiet', negative rates are set to zero and the simulation continues.

`inf.rates`
: One of 'abort', 'warn' or 'quiet', as for`neg.rates`. With 'warn' or 'quiet', infinite rates are left in place, resulting in events happening instantly. This may, in some cases, be desirable!

`stall.time`
: If the all rates within the simulation are zero then only this length of time is allowed to pass before the simulation exits with a 'stalled' status. If `grain` is infinite, then the simulation stalls immediately when all rates are zero.

`extend.proportion`
: This option allows the simulation to continue running for a given proportion of the time to the next speciation. This makes sense when growing a clade to a given number of extant taxa; with the default setting of zero, the resulting tree ends at a bifurcation with zero branch lengths and this option allows the tree to grow (and taxa to go extinct and traits to evolve).

`x`
: A lineage table output from growTree

`...`
: Further arguments to as.

Details

The main idea behind this function (which is still in development) is to provide a flexible framework for simulating tree growth and trait evolution. The user provides expressions for the main arguments (`b`, `d` and `halt`) which act as rules defining speciation and extinction and the ending of the simulation. These can be simple constants, but can also make use of the properties of the environment of the evolving tree. This includes both lineage specific properties (as described in the `lineages` section of the returned value) or properties of the clade as a whole (as described in the `clade` section of the returned value). For example, a extinction rate might increase with lineage age (`d=0.01*lin.age`) or a speciation rate might decrease according to a density dependent process (`b=1 - (nExtantTip/500)`). Halt expressions will typically use clade properties (`halt=clade.age >= 5` or `nTips >= 50`) but could use lineage properties, for example stopping when a trait value hits a certain value (`halt=any(ct1 >= 10)`). It is not permitted to use '==' in a halt function of clade.age because it will allow the simulation to run away if the actual value steps over the test value.

Discrete traits are defined using a matrices of rates for transitions between states for each trait. At present, these are fixed for the duration of a simulation epoch and cannot be set as expressions of tree variables.

Continuous trait evolution currently employs a simple Brownian model, given a starting value and variance per unit time. The traits can have defined co-variance (the simulation uses mvrnorm at present) and can also have a defined mean change, allowing for a directional walk in the trait values. At present, it is not possible for the trait variance to vary according to the internal state of the tree; continuous characters retain the same variance and covariance for the whole of the simulation epoch.

Whilst none of the `halt` rules are TRUE, then the function evaluates the birth, death and discrete trait rates and converts these to waiting times using random variates from a exponential distribution with the calculated rates. These competing waiting times are compared both to each other and the

grain of the simulation, the shortest waiting time is found and the relevant event is then triggered. The winning event is identified in the character vector winnerName in order to allow inheritance rules to differentiate events.

Value

Depending on the value of output.phylo, either an object of class 'phylo' or an object of class 'growTree' with the following structure:

lineages	A data frame with a row for each lineage in the tree. Each row identifies the parent.id and id of the row along with the total age of the lineage (lin.age) and the times at which the lineage was born (birth.time). If the species went extinct (or speciated) then the death.time is recorded and extinct is set to TRUE. Speciating lineages have tip set to FALSE. Each row also records the caic.code of the lineage - this is used as a sorting code for conversion to a 'phylo' object and is a kludge. If traits are defined in the simulation then the values or states are recorded in this table. These are the current values for extant tips and the values at extinction for extinct tips and internal nodes.
clade	A list containing:clade.age, the total age of the simulation; nLin, the total number of lineages; nTip, the total number of tips, differentated into nExtantTip and nExtinctTip.
rules	A list reporting the birth (b), death (d) and stopping (halt) rules and any inheritance rules.
ct.set	If continuous characters were simulated, a list of the ct.start, ct.change and ct.var details provided.
dt.rates	If discrete characters were simulated, a list containing the dt.rates details provided.

Author(s)

David Orme, drawing heavily on discussions with Paul-Michael Agapow.

Examples

```
## see the package vignette for a much fuller discussion of examples.

# A basic 200 tip tree, output as a 'comparative.data' object
    tree <- growTree(halt=200, grain=Inf)
    plot(tree$phy)

# A basic tree of age 4 time units, output as a 'comparative.data' object
    tree <- growTree(halt=expression(clade.age >= 4), grain=Inf)
    plot(tree$phy)
```

`IsaacEtAl` *Example dataset for the caper package*

Description

This data set contains four species-level comparative datasets used in Isaac et al (2005)

Usage

```
data(IsaacEtAl)
```

Format

The datafile contains species level phylogenies and accompanying data frames of nine variables for each of four mammalian orders (Primates, Carnivora, Chiroptera and Marsupialia). The data were published in supplementary material for Isaac et al. (2005) as CAIC format files and text data files and have been converted for use in 'caper'. The data files are incomplete, with some variables having little or no data for some orders.

The variables (all saved as natural log values) are:

species.rich Species richness at the tips - all are set to 1 for use in `macrocaic`

body.mass The average body mass in grams

age.sexual.maturity Age at sexual maturity in months

gestation Gestation length in days

interbirth.interval Interbirth interval in months

litter.size The average number of offspring in a litter

population.density Population density

group.size Number of individuals in a typical group

mass.dimorphism Male mass /female mass

length.dimorphism Male length / female length

References

Isaac, N., Jones, K., Gittleman, J., and Purvis, A. (2005). Correlates of species richness in mammals: Body size, life history, and ecology. American Naturalist, 165(5):600-607.

See Also

caic, pgls

Examples

```
data(IsaacEtAl)
chiroptera  <- comparative.data(chiroptera.tree, chiroptera.data, 'binomial', na.omit=FALSE)
carnivora   <- comparative.data(carnivora.tree, carnivora.data, 'binomial', na.omit=FALSE)
primates    <- comparative.data(primates.tree, primates.data, 'binomial', na.omit=FALSE)
marsupialia <- comparative.data(marsupialia.tree, marsupialia.data, 'binomial', na.omit=FALSE)
```

`macrocaic` *Comparative analysis using independent contrasts on species richness data.*

Description

Macroevolutionary hypotheses about correlates of species richness require testing in a phylogenetic framework in order to avoid phylogenetic autocorrelation. Independent contrasts as described by Felsenstein (1985) are appropriate for explanatory variables in such models but not for species richness as the response variable. This function implements two methods for calculating species richness constrasts described by Agapow and Isaac (2002) and originally implemented in the program MacroCAIC.

Usage

```
macrocaic(formula, data, phy, names.col, macroMethod = "RRD",
          stand.contr = TRUE, robust=Inf, ref.var = NULL, node.depth = NULL,
          macroMinSize = 3, equal.branch.length = FALSE)
```

Arguments

`formula`	A formula describing a linear model predicting species richness.
`data`	A data frame containing the variables to be used in the model.
`phy`	An object of class 'phylo'.
`names.col`	A name identifying a column in `data` that contains the tip labels from `phy`.
`macroMethod`	One of either "RRD" or "PDI" (see Details).
`stand.contr`	A logical flag indicating whether to standardize the contrasts.
`robust`	A threshold value of studentized residuals to exclude from the model.
`ref.var`	Identifies a predictor variable used for determining the direction of contrasts.
`node.depth`	A positive integer greater than 1 used to restrict the model to contrasts with a node depth less than or equal to the specified depth. Tips have a depth of 1.
`macroMinSize`	A positive integer giving the minimum species richness at a node for contrasts to be included in the model.
`equal.branch.length`	If set to 'TRUE' then all branch lengths are set to 2.

Details

The 'macrocaic' function fits a regression to the formula provided using 'crunch' contrasts for continuous explanatory variables and species richness contrasts for the response. The species richness contrasts are either the relative rate difference (RRD) or proportion dominance index (PDI):

$$RRD = \ln\left(\frac{N_1}{N_2}\right)$$

RRD = ln(N_1/N_2)

$$PDI = \left(\frac{N_1}{N_1 + N_2}\right) - 0.5$$

PDI = (N_1/(N_1 + N_2))-0.5

The values N_1 and N_2 are the species richness of the two daughter nodes and N_1 is the species richness of the clade with the larger value of the reference variable. Species richness contrasts are not calculated at polytomies. Nodal values for species richness are calculated as the sum of the richness of the daughter nodes.

Value

A object of class 'caic'.

Author(s)

David Orme

References

Felsenstein, J. (1985). Phylogenies and the comparative method. Am. Nat. 125, 1-15 Agapow, P.-M. and Isaac, N. J. B. (2002) MacroCAIC: correlates of species richness. Diversity & Distributions, 8, 41-43 Isaac, N., Agapow, P., Harvey, P., and Purvis, A. (2003). Phylogenetically nested com- parisons for testing correlates of species richness: A simulation study of continuous variables. Evolution, 57(1):18-26.

See Also

caic-class for 'caic' object structure and methods.

Examples

```
data(IsaacEtAl)
primates <- comparative.data(primates.tree, primates.data, binomial, na.omit=FALSE)
primatesBodySize <- macrocaic(species.rich ~ body.mass, data=primates)
summary(primatesBodySize)
```

pd.calc *Calculate and bootstrap phylogenetic diversity measurements.*

Description

These functions calculate various phylogenetic diversity measures for either a given set of nodes on a tree or for a randomly chosen set of nodes of a given size. The ed.calc function calculates a related species-level measurement of evolutionary distinctness.

Usage

```
pd.calc(cm, tip.subset = NULL, method = "TBL", root.edge=FALSE)
pd.bootstrap(cm, ntips, reps = 1000, method = "TBL", tip.weights = NULL)
ed.calc(cm, polytomy.cf=c("isaac","mooers","none"))
```

Arguments

cm	A object of class 'clade matrix'. Alternatively an object of class 'phylo', which will be converted to a clade.matrix.
tip.subset	An optional vector identifying the subset of tips to use for PD calculations. If no tip.subset is provided the method is applied to the complete phylogeny [Hmm.. this might be undesirable]. Can either be a character vector, in which case the elements are matched against tip labels, or a vector of positive integers in the range 1 to the number of tips, in which case the tips with those numbers are used.
method	One of 'TBL', 'MST', 'UEH', 'SBL', defaulting to 'TBL'. See details.
root.edge	Logical indicating whether to include the root edge length in calculations, defaulting to FALSE.
ntips	A single integer giving the number of tips to be selected.
reps	The number of replicate values to calculate.
tip.weights	A numeric vector containing weights for all the tips in the phylogeny. Each element must be named in order to match weights to the tips.
polytomy.cf	Which correction factor to use for calculating ED at polytomies. One of 'isaac', 'mooers' or 'none'.

Details

There are five implemented PD measures:

Total Branch Length (TBL) The sum of all the edge lengths in the subtree given by the tip subset. This measure can be partitioned into the two next measures.

Shared Branch Length (SBL) The sum of all edges in the subtree that are shared by more than one tip.

Unique Evolutionary History (UEH) The sum of the edge lengths that give rise to only one tip in the subtree.

Length of tip branch lengths (TIPS) Length of tip branch lengths (TIPS)Unlike UEH, this measure does not use the unique paths to each tips on the **subtree** and instead gives the sum of the unique branches leading to the tips on the **complete tree**.

Minimum Spanning Tree (MST) The sum of the lengths of the edges for the smallest tree that links the subset tips, excluding any edges below the node of the most recent common ancestor.

These options are illustrated in the caper package vignette. The pd.calc function returns the PD value for a given set of tips, whereas the pd.bootstrap function returns a vector of PD values for randomly selected sets of tips of a given size.

The ed.calc function returns the evolutionary distinctness (ED) metric (Isaac et al, 2007) for the tips of a given phylogeny. The idea behind the ED measure is that the evolutionary history of each branch is shared equally between all tips descending from that branch. Each branch therefore has a per-tip values of the branch length divided by the number of descendants and the ED value for a tip is the sum of those per-tip contributions over the path to the root of the phylogeny. Polytomies inflate apparent ED since the branches of a properly resolved polytomy must be shorter than the branch lengths on the unresolved polytomy. The function provides two correction factors for this: 'isaac' uses a correction factor calibrated from simulations and 'mooers' uses empirical predictions from a pure birth model.

Value

Both pd.calc and pd.bootstrap return a vector containing either a single value for the phylogenetic diversity of a given set of tips or a vector of length 'nrep' containing the pd values for a random set of tips of a given size. The method used is stored in the 'pd.method' attribute of the vector.

The ed.calc function returns a list containing:

branch A data frame of the ED contributions arising from each branch.

spp A data frame of the summed ED contributions for each species.

Author(s)

David Orme, Gavin Thomas, Nick Isaac

References

Faith, DP, Isaac, N. J. B., Turvey, S. T., Collen, B., Waterman, C., and Baillie, J. E. M. (2007). Mammals on the edge: Conservation priorities based on threat and phylogeny. Plos One, 2(3):e296

Examples

```
treeString <- paste('((((A:1,B:1):1.5,C:2.5):0.5,(D:0.6,E:0.6):2.4):0.5,',
                    '((F:1.9,G:1.9):0.8,(H:1.6,I:1.6):1.1):0.8):0.2;' ,sep='')
tre <- read.tree(text=treeString)
clmat <- clade.matrix(tre)
tips <- c("A","C","D","E","G","H")
pd.calc(clmat, tip.subset=tips)
pd.calc(clmat, tip.subset=c(1,3,4,5,7,8))
pd.calc(clmat, tip.subset=tips, root.edge=TRUE)

pd.bootstrap(clmat, ntips=6, reps=1000, method='TBL')
```

```
data(IsaacEtAl)
primatesCM <- clade.matrix(primates.tree)
primatesED <- ed.calc(primatesCM)
```

---

perissodactyla *Example dataset for the CAIC package*

Description

This is a comparative dataset on Perissodactyla taken from the examples include with the original CAIC program.

Usage

```
data(shorebird)
```

Format

The datafile contains a phylogeny (`perissodactyla.tree`) of 18 perissodactyl species as a 'phylo' object from the `ape` library. The tip names are the binomial names of the species. The file also contains a data frame (`perissodactyla.data`) of variables 5 variables for 13 of those species:

Binomial The species binomial name.

log.female.wt Log female weight

log.gestation.length Log gestation length

log.neonatal.wt Log neonatal weight

Territoriality A factor indicating whether or not the species displays territorial behaviour.

The dataset is incomplete - it does not include data for each species in the phylogeny and contains missing values. See the examples for the behaviour of the 'comparative.data' function in handling missing data.

References

Purvis, A. and Rambaut, A. (1995). Comparative Analysis by Independent Contrasts (CAIC) User's Guide.

See Also

caic, pgls

Examples

```
data(perissodactyla)
# default behaviour is to omit incomplete data rows
(perisso <- comparative.data(perissodactyla.tree, perissodactyla.data, Binomial))
# but this can be turned off
(perisso <- comparative.data(perissodactyla.tree, perissodactyla.data, Binomial, na.omit=FALSE))
na.omit(perisso)
```

pgls *Phylogenetic generalized linear models*

Description

Fits a linear model, taking into account phylogenetic non-independence between data points. The strength and type of the phylogenetic signal in the data matrix can also be accounted for by adjusting branch length transformations (lambda, delta and kappa). These transformations can also be optimised to find the maximum likelihood transformation given the data and the model.

Usage

```
pgls(formula, data, lambda = 1.0, kappa = 1.0,  delta= 1.0, param.CI = 0.95,
 control = list(fnscale=-1), bounds = NULL)
pgls.likelihood(optimPar, fixedPar, y, x, V, optim.output=TRUE, names.optim=NULL)
pgls.blenTransform(V, fixedPar)
```

Arguments

formula	A model formula
data	A 'comparative.data' object containing the covariance matrix and data to be used in the model.
lambda	A value for the lambda transformation.
kappa	A value for the kappa transformation.
delta	A value for the delta transformation.
param.CI	A p value used to calculate confidence intervals.
control	A list of control parameters for the optim function.
bounds	A list of bounds to use for branch length transformations (see Details).
optimPar	A named vector of branch length parameters to be optimised to find the maximum likelihood value.
fixedPar	A named vector of fixed values for branch length parameters.
y	A column matrix of the model response.
x	The design matrix of the model.
V	A phylogenetic covariance matrix.
optim.output	A logical value. If true then 'pgls.likelihood' returns only the likelihood value for use in the 'optim' function.
names.optim	The name of a single parameter being optimised. This is only required for estimating parameter confidence intervals, where the function 'uniroot' strips names from vectors.

Details

This function fits a linear model controlling for the non-independence between cases resulting from phylogenetic structure in the data. The stucture of the phylogenetic signal can be controlled by altering the parameters lambda, delta and kappa (see the 'caper' vignette for details). The implementation of the method is currently as described in Freckleton et al (2002).

The branch length transformations can be optimised between bounds using maximum likelihood by setting the value for a transformation to 'ML'. The default bounds are: lambda = c(1e-6,1), kappa = c(1e-6,3) and delta=c(1e-6,3). These defaults may be overridden by passing a named list with new elements to the bounds argument - only the bounds to be changed need to be provided (e.g. bounds=list(lambda=c(0,3))).

The 'pgls.likelihood' and 'pgls.blenTransform' methods are not primarily intended to be called by users. The 'pgls.likelihood' function provides a general method to calculate the likelihood of a model, given the covariance matrix, response, design matrix and branch length parameters.

Value

The 'pgls' function returns an object of class `pgls` containing the following:

"na.action" "param.CI"

`call`	The original call to the 'pgls' function
`model`	A summary of the fitted model containing:
`formula`	The model formula supplied.
`data`	The comparative data object provided.
`dname`	The name of the comparative data object.
`logLikY`	The log likelihood of the response variable given the model.
`RMS`	The residual mean square variance in the model.
`RSSQ`	The residual sum of squares from the model.
`NMS`	The null mean square variance for the model.
`NSSQ`	The null sum of squares for the response.
`aic`	The AIC score of the model
`aicc`	The AICc score of the model, correcting for the number of cases and parameters estimated
`n`	The number of rows of data used in fitting the model
`k`	The number of parameter estimates
`sterr`	The standard errors of the parameter estimates
`Vt`	The phylogenetic covariance matrix used in the model, with branch length transformations applied.
`fitted`	The predicted values
`residuals`	The non-phylogenetic residuals
`phyres`	The phylogenetic residuals
`x`	The design matrix of the model

varNames	The variables include in the model.
y	The response of the model.
namey	The name of the response variable.
param	A named numeric vector of length three giving the branch length transformations used in the model.
mlVals	A named logical vector of length three indicating which branch length values in 'param' are maximum likelihood estimates.
bounds	The bounds on branch length parameter estimates used in the model.
param.CI	A named list of length three giving confidence intervals and the p values at the parameter bounds for optimised branch length transformations. Fixed parameters will have a NULL entry in this list.
na.action	A named vector identifying any rows of missing data excluded from the model.

Warning

The model is fitted using a data frame reduced to complete row cases to eliminate missing values. In order to ensure that the models fitted using different subsets of the data are comparable, the whole data frame `data` is reduced to complete cases. In the future, a scope argument may be provided to control this but at present the data frame should be reduced to only those variables used in the maximal model in order to avoid prevent redundant variables causing rows to be dropped unnecessarily.

Author(s)

Rob Freckleton; David Orme

References

R. P. Freckleton, P. H. Harvey, and M. Pagel. Phylogenetic analysis and comparative data: A test and review of evidence. American Naturalist, 160:712-726, 2002.

See Also

`pgls.profile`, `anova.pgls`, `summary.pgls`

Examples

```
data(shorebird)
shorebird <- comparative.data(shorebird.tree, shorebird.data, Species, vcv=TRUE, vcv.dim=3)
mod1 <- pgls(log(Egg.Mass) ~ log(M.Mass) * log(F.Mass), shorebird, lambda='ML')
mod2 <- pgls(log(Egg.Mass) ~ log(M.Mass), data=shorebird, lambda='ML', delta='ML')
```

pgls-methods *Generic model methods for 'pgls' models.*

Description

These are simple summary methods, accessor functions and summary and print methods for 'pgls' models.

Usage

```
## S3 method for class 'pgls'
coef(object, ...)
## S3 method for class 'pgls'
residuals(object, phylo = FALSE, ...)
## S3 method for class 'pgls'
fitted(object, ...)
## S3 method for class 'pgls'
predict(object, newdata, ...)
## S3 method for class 'pgls'
summary(object, ...)
## S3 method for class 'pgls'
print(x, digits = max(3, getOption("digits") - 3), ...)
## S3 method for class 'summary.pgls'
print(x, digits = max(3, getOption("digits") - 3), ...)
## S3 method for class 'pgls'
nobs(object, ...)
```

Arguments

object	An object of class 'pgls'.
x	An object of class 'pgls'.
phylo	Return phylogenetically corrected residuals or ordinary residuals (see details).
newdata	Alternative data for predicting from 'pgls' models.
digits	Number of digits to show in summary methods.
...	Further arguments to methods.

Details

Phylogenetically corrected residuals from 'pgls' models [TODO].

Note that the r^2 values reported by `summary.pgls` have a specific interpretation. `pgls` fits the intercept-only model for the data using _exactly_ the same covariance matrix (phylogeny plugged through any branch length transformations) as the fitted model to get a null model. The r-squared and adjusted r-squared that are reported therefore hold the covariance matrix constant, so show percentage of variance explained between a null model and the actual model given that precise model of trait change.

The actual ML null model for the data (optimising the BL transformation independently) might be different from this - but then the r squared values confound change in explanatory power from changing the model parameters and from changing the trait model.

Value

The 'summary' method returns an object of class 'summary.pgls' containing:

`call`	The original function call creating the model.
`df`	A vector of the degrees of freedom used to estimate parameters and the residual degrees of freedom.
`sigma`	The square root of the estimated variance of the random error.
`residuals`	The phylogenetically corrected residuals.
`coefficients`	A table of model coefficient, standard errors and t values.
`param`	A vector of branch length parameters used in the model.
`mlVals`	A vector showing which branch length parameters have been optimised.
`param.CI`	A list of length three containing confidence intervals and p values on parameter bounds for each parameter.
`fstatistic`	A vector of the F value, numerator and denominator degrees of freedom for the model.
`r.squared`	The r^2 for the model.
`adj.r.squared`	The adjusted r^2 for the model.

Author(s)

Rob Freckleton, David Orme

See Also

pgls

Examples

```
data(shorebird)
shorebird <- comparative.data(shorebird.tree, shorebird.data, Species, vcv=TRUE, vcv.dim=3)
mod1 <- pgls(log(Egg.Mass) ~ log(M.Mass) * log(F.Mass), shorebird)
print(mod1)

mod1.sum <- summary(mod1)
print(mod1.sum)
```

pgls.profile *Likelihood profiles and confidence intervals for 'pgls' models.*

Description

These functions create likelihood profiles for branch length transformations in phylogenetic generalised least squares models and fit confidence intervals to estimated branch length parameters.

Usage

```
pgls.profile(pgls, which = c("lambda", "kappa", "delta"), N = 50, param.CI = NULL)
pgls.confint(pgls, which=c('lambda','kappa','delta'), param.CI=0.95)
## S3 method for class 'pgls.profile'
plot(x, ...)
```

Arguments

pgls	A pgls object.
which	A choice of which branch length transformation ('lambda', 'kappa' or 'delta') to use.
N	The number of points used to profile the likelihood
param.CI	A p value used to add confidence intervals to a likelihood profile for a parameter.
x	A 'pgls.profile' object to plot.
...	Further arguments to plot functions.

Details

The 'pgls.profile' function calculates the likelihood of a 'pgls' model under different values of branch length transformations. A single parameter is chosen from 'lambda', 'kappa' or 'delta' to be profiled and the model likelihood is calculated at 'N' equally spaced points between the parameter bounds used in the model. If the model contains a maximum likelihood estimate of the parameter (or if param.CI is not null) then the resulting 'pgls.profile' object will contain estimated confidence intervals.

Only one parameter is profiled at a time and the other branch length parameters will be held at the fixed or ML estimates used to fit the model. The 'pgls.confint' function is used by either 'pgls' or 'pgls.profile' to find confidence intervals around a maximum likelihood estimate of a given branch length. The model must contain an ML estimate of the parameter for confidence intervals to be calculated.

The plot method simply draws an annotated profile plot, showing the location of the ML estimate and confidence intervals if present.

Value

The 'pgls.profile' function returns a list containing:

`x`	Parameter values at which the likelihood has been calculated.
`logLik`	The likelihood value at each value.
`which`	The parameter being profiled.
`pars`	The value of the other fixed parameters.
`dname`	The name of the 'comparative.data' object used to fit the model.
`formula`	The formula of the model being profiled

If the model contains an ML estimate of the parameter being profiled, then the 'pgls.profile' object will also contain the output of 'pgls.confint':

`opt`	The maximum likelihood value of the parameter.
`bounds.val`	The values of the bounds on the parameter.
`bounds.p`	The p value of the likelihood at the bounds, given the ML value.
`ci.val`	The values of the parameter at the confidence intervals.
`ci`	The confidence interval value used.

Author(s)

David Orme

See Also

pgls

Examples

```
data(shorebird)
shorebird <- comparative.data(shorebird.tree, shorebird.data, Species, vcv=TRUE, vcv.dim=3)
mod <- pgls(log(Egg.Mass) ~ log(M.Mass), shorebird, lambda='ML')
mod.l <- pgls.profile(mod, 'lambda')
plot(mod.l)
pgls.confint(mod, 'lambda')
```

phylo.d *Calculates the phylogenetic D statistic*

Description

Calculates the D value, a measure of phylogenetic signal in a binary trait, and tests the estimated D value for significant departure from both random association and the clumping expected under a Brownian evolution threshold model.

Usage

```
phylo.d(data, phy, names.col, binvar, permut = 1000, rnd.bias=NULL)
## S3 method for class 'phylo.d'
print(x, ...)
## S3 method for class 'phylo.d'
summary(object, ...)
## S3 method for class 'phylo.d'
plot(x, bw=0.02, ...)
```

Arguments

data	A 'comparative.data' or 'data.frame' object.
phy	An object of class 'phylo', required when data is not a 'comparative.data' object.
names.col	A name specifying the column in 'data' that matches rows to tips in 'phy', required when data is not a 'comparative.data' object.
binvar	The name of the variable in data holding the binary variable of interest.
permut	Number of permutations to be used in the randomisation test.
rnd.bias	An optional name of a variable in data holding probability weights to bias the generation of the random distribution. See 'destails'
x	An object of class 'phylo.d'
object	An object of class 'phylo.d'
bw	The bandwidth to be used for the density plots
...	Further arguments to print and summary methods

Details

The sum of changes in estimated nodal values of a binary trait along edges in a phylogeny (D) provides a measure of the phylogenetic signal in that trait (Fritz and Purvis, 2010). If a trait is highly conserved, with only a basal division between two clades expressing either trait value, then the only change will be along the two daughters at the root. This will give a summed value of 1: the two differences between the root nodal value of 0.5 and the ancestors of the 1 and 0 clades. In contrast, if the trait is labile, more differences will be observed and the sum will be higher.

This function calculates the observed D for a binary trait on a tree and compares this to the value of D found using an equal number of simulations under each of two models:

Phylogenetic randomness Trait values are randomly shuffled relative to the tips of the phylogeny and D is calculated.

Brownian threshold model A continuous trait is evolved along the phylogeny under a Brownian process and then converted to a binary trait using a threshold that reproduces the relative prevalence of the observed trait.

The value of D depends on phylogeny size - more sister clades yield higher sums - and so the means of the two sets of simulated data are used as calibrations to scale both observed and simulated values of D to set points of 0 (as phylogenetically conserved as expected under a Brownian threshold model) and 1 (random). The value of D can be both smaller than 0 (highly conserved) and greater than 1 (overdispersed) and the distributions of scaled D from the simulations are used to assess the significance of the observed scaled D. The `plot` method generates density plots of the distributions of the two simulations relative to the observed D value.

`rnd.bias` is passed to `sample` as the `prob` argument to weight the random shuffles of the observed trait. The weights are not checked for validity.

Value

Returns an object of class 'phylo.d', which is a list of the following:

`DEstimate`	The estimated D value
`Pval1`	A p value, giving the result of testing whether D is significantly different from one
`Pval0`	A p value, giving the result of testing whether D is significantly different from zero
`Parameters`	A list of the Observed, MeanRandom and MeanBrownian sums of sister-clade differences
`Permutations`	A list with elements random and brownian, containing the sums of sister-clade differences from random permutations and simulations of Brownian evolution under a threshold model
`NodalVals`	A list with the elements observed, random and brownian, containing the nodal values estimated for the observed trait and permutations. The values are as matrices with rows labelled by the node names in the comparative data object.
`binvar`	The binary variable used
`phyName`	The name of the phylogeny object used
`dsName`	The name of the dataframe used
`nPermut`	The number of permutations used
`rnd.bias`	If a bias was introduced to the calculation of the random distribution, the bias used, else `NULL`

Author(s)

Susanne Fritz <Susanne.Fritz@senckenberg.de> and David Orme

References

Fritz, S. A. and Purvis, A. (2010). Selectivity in mammalian extinction risk and threat types: a new measure of phylogenetic signal strength in binary traits. Conservation Biology, 24(4):1042-1051.

Examples

```
data(BritishBirds)
BritishBirds <- comparative.data(BritishBirds.tree, BritishBirds.data, binomial)
redPhyloD <- phylo.d(BritishBirds, binvar=Red_list)
print(redPhyloD)
plot(redPhyloD)
```

`phylo.d.subset` *Calculates the phylogenetic D statistic across clades within a phylogeny*

Description

Calculates the D value, a measure of phylogenetic signal in a binary trait, and tests the estimated D value for significant departure from both random association and the clumping expected under a Brownian evolution threshold model. Does this across clades within a phylogeny.

Usage

```
phylo.d.subset(data, phy, names.col, binvar, permut = 1000, rnd.bias=NULL,
           min.tips=1, max.tips=length(data$phy$tip.label), min.nodes=1,
   max.nodes=data$phy$Nnode, verbose=FALSE)
## S3 method for class 'phylo.d.subset'
print(x, ...)
## S3 method for class 'phylo.d.subset'
summary(object, ...)
```

Arguments

`data`	A 'comparative.data' or 'data.frame' object.
`phy`	An object of class 'phylo', required when data is not a 'comparative.data' object.
`names.col`	A name specifying the column in 'data' that matches rows to tips in 'phy', required when data is not a 'comparative.data' object.
`binvar`	The name of the variable in `data` holding the binary variable of interest.
`permut`	Number of permutations to be used in the randomisation test.
`rnd.bias`	An optional name of a variable in `data` holding probability weights to bias the generation of the random distribution. See 'destails'
`verbose`	Logical; do you want to know how many clades are being assessed, and see when each is being assessed?

min.tips	The minimum number of tips a clade should have for it to have a D value calculated. Defaults to 1 (i.e. no limit).
max.tips	The maximum number of species a clade should have for it to have a D value calculated. Defaults to the number of species in the whole phylogeny (i.e. no limit).
min.nodes	The minimum number of nodes a clade should have for it to have a D value calculated. Defaults to 1 (i.e. no limit).
max.nodes	The maximum number of nodes a clade should have for it to have a D value calculated. Defaults to the number of nodes in the whole phylogeny (i.e. no limit).
x	An object of class 'phylo.d.subset'
object	An object of class 'phylo.d.subset'
...	Further arguments to print and summary methods

Details

A wrapper function for phylo.d, calculating D values for clades within a given dataset. These clades can be filtered according to the number of species and nodes using the arguments above. See phylo.d for more details on the method itself.

Any clades for which there is no variation in the binary variable have NA values for all of the below slots.

Value

Returns an object of class 'phylo.d.subset', which is a list of the following:

raw	A list of the raw output from phylo.d for each clade
DEstimate	A vector of the estimated D values
Pval1	A vector of p values, giving the result of testing whether D is significantly different from one, for each clade
Pval0	A vector of p values, giving the result of testing whether D is significantly different from zero, for each clade
phy.depth	A numeric vector giving the age of the clade for which each value was calculated

Author(s)

Susanne Fritz (SFritz@bio.ku.dk), Will Pearse and David Orme

References

Fritz, S. A. and Purvis, A. (2010). Selectivity in mammalian extinction risk and threat types: a new measure of phylogenetic signal strength in binary traits. Conservation Biology, 24(4):1042-1051.

Examples

```
data(BritishBirds)
BritishBirds <- comparative.data(BritishBirds.tree, BritishBirds.data, binomial)
# Look at big clades only
## Not run:
bigClades <- phylo.d.subset(BritishBirds, binvar=Red_list, verbose=TRUE, min.tips=10, min.nodes=5)
print(bigClades)

## End(Not run)
```

`plot.pgls` *Diagnostic plots for 'pgls' models.*

Description

The function generates four diagnostics plots for 'pgls' models.

Usage

```
## S3 method for class 'pgls'
plot(x, ...)
```

Arguments

`x`	An object of class 'pgls'.
`...`	Additional arguments to plot functions.

Details

The first two plots show the fit of the phylogenetic residuals from the model to a normal distribution: a density plot of the residuals and a normal Q-Q plot. The second two plots scatterplots show pattern in the distribution of the fitted values against the observed and residual values.

Author(s)

Rob Freckleton, David Orme

See Also

pgls

Examples

```
data(shorebird)
shorebird <- comparative.data(shorebird.tree, shorebird.data, Species, vcv=TRUE, vcv.dim=3)
mod1 <- pgls(log(Egg.Mass) ~ log(M.Mass) * log(F.Mass), shorebird)
par(mfrow=c(2,2))
plot(mod1)
```

shorebird *Example dataset for the caper package*

Description

This is a comparative dataset on the evolution of shorebird egg size taken from Lislevand and Thomas (2006).

Usage

```
data(shorebird)
```

Format

The datafile contains a phylogeny (`shorebird.tree`) of 71 shorebird species as a 'phylo' object from the ape library. The tip names are the binomial names of the species. The file also contains a data frame (`shorebird.data`) of 71 complete cases for those species. The data frame contains six variables:

Species The species binomial name.

M.Mass The adult male mass body mass in grams.

F.Mass The adult female mass body mass in grams.

Egg.Mass The fresh egg mass in grams.

Cl.size The mean clutch size

Mat.syst The mating system, as a three level factor: monogamous (MO), polygynous (PO) or polyandrous (PA).

References

Lislevand, T and Thomas, G. H. (2006) Limited male incubation ability and the evolution of egg size in shorebirds. Biology Letters 2, 206 - 208

See Also

crunch, pgls

summary.caic *Summarize a crunch, brunch or macrocaic analysis*

Description

The summary method simply returns the linear model summary from the 'caic' object. The print method prints some basic information about the analysis followed by the model summary.

Usage

```
## S3 method for class 'caic'
summary(object, ...)
## S3 method for class 'caic'
print(x,...)
```

Arguments

object	An object of class 'caic'.
x	An object of class 'caic'.
...	Arguments to be passed to 'summary.lm'.

Value

The summary method returns an object of class 'summary.lm'.

Author(s)

David Orme

See Also

crunch,brunch, link{macrocaic}

Examples

```
data(shorebird)
shorebird <- comparative.data(shorebird.tree, shorebird.data, Species)
crunchMod <- crunch(Egg.Mass ~ F.Mass + M.Mass, data=shorebird)
print(crunchMod)
summary(crunchMod)
```

syrphidae *The syrphidae dataset of Katzourakis et al. 2001*

Description

A genus level phylogeny of the Syrphidae (sawflies) along with data on the species richness of each genus.

Usage

```
data(syrphidae)
```

Format

A 'phylo' object (`syrphidaeTree`) containing a phylogeny of 204 genera of sawflies and a data frame (`syrphidaeRich`) of the species richness of each genus.

References

Katzourakis, A., Purvis, A., Azmeh, S., Rotheray, G., and Gilbert, F. (2001). Macroevolution of hoverflies (Diptera : Syrphidae): the effect of using higher-level taxa in studies of biodiversity, and correlates of species richness. Journal of Evolutionary Biology, 14:219-227.

See Also

fusco.test

VCV.array *Create a 2D or 3D variance-covariance matrix from a phylogeny*

Description

The function turns a phylogeny into a variance-covariance matrix, as in the function `vcv.phylo` in the 'ape' package but can also return a 3D array retaining the individual branch lengths contributing to the shared branch lengths. This is useful for handling some branch length transformations, such as kappa, and has a lower overhead than repeatedly calling `vcv.phylo` on a phylogeny after transforming the vector of edge lengths.

Usage

```
VCV.array(phy, dim=2, compact=TRUE)
```

Arguments

`phy`	An object of class 'phylo'.
`dim`	Either 2, for a standard VCV matrix, or 3, for an array of branch lengths.
`compact`	A logical vector indicating the form to use for a 3D array.

Details

The compact form of the 3D array uses a shortened third dimension, which is only long enough to hold the maximum number of shared branches between root and tip for each pair of tips. Zeros are used to pad out this depth vector for tip pairs with shorter paths. The non-compact form returns 3D array showing, for each pair of tips and each node in the tree, either 0 if the node is not shared or the appropriate edge length if the node is shared. Note that, for maximally unbalanced trees, the size of the two forms will be identical.

The algorithm for the noncompact form is faster than for the compact form but it has very high memory overheads on big trees. The 2 dimensional algorithm is at least twice as fast as `vcv.phylo` on trees up to 2500 tips.

The `apply` function can be easily used to collapse the array down to a standard VCV matrix, as in the example.

Value

When dim = 2, a variance covariance matrix of class 'VCV.array' of dimension nTips by nTips with dimnames set from the tip labels of the phylogeny.

When dim = 3, a 3 dimensional array of class 'VCV.array' with dimensions of the number of taxa in the phylogeny for the rows and columns and either the maximum number of branches on the root to tip path or the number of internal nodes as the depth, depending on the setting of `compact`. The rows and columns are named using the tip labels from the phylogeny and the depth only named with node numbers if `compact` is TRUE.

Author(s)

David Orme

See Also

`vcv.phylo`, `pgls`

Examples

```
tree <- rcoal(8)
tree.VCV <- vcv.phylo(tree)
tree.VCVA <- VCV.array(tree)

# reconstruct a simple VCV array
tree.VCVA.reduced <- apply(tree.VCVA, c(1,2), sum, na.rm=TRUE)

# minimal differences between the two
all((tree.VCVA.reduced - tree.VCV) < 1e-10)

# a kappa transformation of 0.5
apply(tree.VCVA ^ 0.5, c(1,2), sum, na.rm=TRUE)
```

Index