Package 'RRphylo'

April 6, 2018

Type Package

Title Phylogenetic Ridge Regression Methods for Comparative Studies

Version 1.0.0

Author Pasquale Raia, Silvia Castiglione, Carmela Serio, Alessandro Mondanaro, Marina Melchionna, Mirko Di Febbraro, Antonio Profico, Francesco Carotenuto

Maintainer Pasquale Raia <pasquale.raia@unina.it>

Description Functions for phylogenetic analysis (Castiglione et al, 2017 <doi:10.1111/2041-210X.12954>). The functions perform the estimation of phenotypic evolutionary rates, identification of phenotypic evolutionary rate shifts, quantification of direction and size of evolutionary change in multivariate traits, and the computation of ontogenetic shape vectors.

License GPL-2

Encoding UTF-8

LazyData true

Depends R (>= 3.4.0)

Imports ape, phytools, geiger, stats4, foreach, doParallel, pvclust, mvMORPH, lmtest, parallel, phangorn, smatr, rlist, scales, R.utils

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2018-04-06 11:37:52 UTC

R topics documented:

`RRphylo-package` *Phylogenetic Ridge Regression Methods for Comparative Studies*

Description

RRphylo provides tools for phylogenetic comparative analysis. The main functions allow estimation of phenotypic evolutionary rates, identification of shifts in rate of evolution, quantification of direction and size of evolutionary change of multivariate traits, and computation of species ontogenetic vectors. Additionally, there are functions for simulating phenotipic data, manipulating phylogenetic trees, and retrieving information from phylogenies. Finally, there are functions to plot and test rate shifts at particular nodes.

The complete list of functions can be displayed with `library(help = RRphylo)`

Author(s)

Pasquale Raia, Silvia Castiglione, Carmela Serio, Alessandro Mondanaro, Marina Melchionna, Mirko Di Febbraro, Antonio Profico, Francesco Carotenuto

References

Castiglione, S., Tesone, G., Piccolo, M., Melchionna, M., Mondanaro, A., Serio, C., Di Febbraro, M., & Raia, P.(2018). A new method for testing evolutionary rate variation and shifts in phenotypic evolution. *Methods in Ecology and Evolution*, in press.doi:10.1111/2041-210X.12954

Piras, P., Silvestro, D., Carotenuto, F., Castiglione, S., Kotsakis, A., Maiorino, L., Melchionna, M., Mondanaro, A., Sansalone, G., Serio, C., Vero, V.A., & Raia, P. (2018). Evolution of the sabertooth mandible: A deadly ecomorphological specialization. *Palaeogeography, Palaeoclimatology, Palaeoecology*, in press

`angle.matrix` *Ontogenetic shape vectors analysis*

Description

This function computes and compares ontogenetic vectors among species in a tree.

Usage

```
angle.matrix(RR,node,Y=NULL,select.axes=c("no","yes"),
type=c("phenotypes","rates"),cova=NULL)
```

Arguments

`RR`	an object produced by `RRphylo`.
`node`	the number identifying the most recent common ancestor to all the species the user wants ontogenetic vectors be computed.
`Y`	multivariate trait values at tips.
`select.axes`	if `"yes"`, `'Y'` variables are individually regressed against developmental stages and only significant variables are retained to compute ontogenetic vectors. All variables are retained otherwise.
`type`	specifies weather to perform the analysis on phenotypic (`"phenotypes"`) or rate (`"rates"`) vectors.
`cova`	the covariate to be indicated if its effect on rate values must be accounted for. Contrary to `RRphylo`, `'cova'` needs to be as long as the number of tips in the tree. As the covariate only affects rates computation, there is no covariate to provide when `type = "phenotypes"`.

Details

The `angle.matrix` function takes as objects a phylogenetic tree (retrieved directly from an `RRphylo` object), including the different ontogenetic stages of each species as polytomies. Names at tips must be written as species ID and stage number separated by the underscore. The `RRphylo` object `angle.matrix` is fed with is just used to extract the dichotomized version of the phylogeny. This is necessary because node numbers change randomly at dichotomizing non-binary trees. However, when performing `angle.matrix` with the covariate the `RRphylo` object must be produced without accounting for the covariate. Furthermore, as the covariate only affects the rates computation, it makes no sense to use it when computing vectors for phenotypic variables. Once angles and vectors are computed, `angle.matrix` performs two tests by means of standard major axis (SMA) regression. For each species pair, the "biogenetic test" verifies whether the angle between species grows during development, meaning that the two species becomes less similar to each other during growth. The "paedomorphosis test" tells whether there is heterochronic shape change in the data. Under paedomorphosis, the adult stages of one (paedomorphic) species will resemble the juvenile stages of the other (peramorphic) species. The test regresses the angles formed by the shapes at different ontogenetic stages of a species to the shape at the youngest stage of the other in the pair, against age. Then, it tests whether the two regression lines (one per species) have different slopes,

and whether they have different signs. If the regression lines point to different directions, it means that one of the two species in the pair resembles, with age, the juveniles of the other, indicating paedomorphosis. Ontogenetic vectors of individual species are further computed, in reference to the MRCA of the pair, and to the first stage of each species (i.e. intraspecifically). Importantly, the size of the ontogenetic vectors of rates tell whether the two species differ in terms of developmental rate, which is crucial to understand which process is behind paedomorphosis, where it applies.While performing the analysis, the function prints messages on-screen informing about tests results. If `select.axes = "yes"`, informs the user about which phenotypic variables are used.

Value

A list containing 4 objects:

1. **$regression.matrix** a 'list' including 'angles between species' and 'angles between species to MRCA' matrices for all possible combinations of species pairs from the two sides descending from the MRCA. For each matrix, corresponding biogenetic and paedomorphosis tests are reported.
2. **$angles.2.MRCA.and.vector.size** a 'data.frame' including angles between the resultant vector of species and the MRCA and the size of the resultant vector computed from species to MRCA, per stage per species.
3. **$ontogenetic.vectors2MRCA** a 'data.frame' including angle, size, and corresponding x and y components, of ontogenetic vectors computed between each species and the MRCA. For both angle and size, the p-value for the difference between species pairs is reported.
4. **$ontogenetic.vectors.to.1st.stage** a 'list' containing:
   - $matrices: for all possible combinations of species pairs from the two sides descending form the MRCA, the upper triangle of the matrix contains the angles between different ontogenetic stages for the first species. The same applies to the lower triangle, but for the second species.
   - $vectors: for all possible combinations of species pairs from the two sides descending form the MRCA, angles and sizes of ontogenetic vectors computed to the first stage of each species. For both, the p-value for the difference between the species pair is reported.

Author(s)

Pasquale Raia, Silvia Castiglione, Carmela Serio, Alessandro Mondanaro, Marina Melchionna, Mirko Di Febbraro, Antonio Profico, Francesco Carotenuto

Examples

```
  data("DataApes")
  DataApes$PCstage->PCstage
  DataApes$Tstage->Tstage
  DataApes$CentroidSize->CS


  RRphylo(tree=Tstage,y=PCstage)->RR
# Case 1. without accounting for the effect of a covariate

 # Case 1.1 selecting shape variables that show significant relationship with age
```

```
  # on phenotypic vectors
    angle.matrix(RR,node=72,Y=PCstage,select.axes="yes",type="phenotypes")
  # on rates vectors
    angle.matrix(RR,node=72,Y=PCstage,select.axes="yes",type="rates")

 # Case 1.2 using all shape variables
  # on phenotypic vectors
    angle.matrix(RR,node=72,Y=PCstage,select.axes="no",type="phenotypes")
  # on rates vectors
    angle.matrix(RR,node=72,Y=PCstage,select.axes="no",type="rates")


# Case 2. accounting for the effect of a covariate (on rates vectors only)

 # Case 2.1 selecting shape variables that show significant relationship with age
   angle.matrix(RR,node=72,Y=PCstage,select.axes="yes",type="rates", cova=CS)


 # Case 2.2 using all shape variables
   angle.matrix(RR,node=72,Y=PCstage,select.axes="no",type="rates",cova=CS)
```

`DataApes` *Example dataset*

Description

Geometric morphometrics shape data regarding Apes' facial skeleton and Apes phylogentic trees (*Profico et al. 2017*).

Usage

```
data(DataApes)
```

Format

A list containing:

$PCstage A data frame containing 38 shape variables for Apes' facial skull at different ontogenetic stages.

$PCadult A data frame containing 3 shape variables for Apes' facial skull.

$Tstage Phylogenetic tree of Apes including the different ontogenetic stages of each species as polytomies.

$Tadult Phylogenetic tree of Apes.

$CentroidSize numeric vector of Centroid Size values of 'PCstage'.

Author(s)

Pasquale Raia, Silvia Castiglione, Carmela Serio, Alessandro Mondanaro, Marina Melchionna, Mirko Di Febbraro, Antonio Profico, Francesco Carotenuto

References

Profico, A., Piras, P., Buzi, C., Di Vincenzo, F., Lattarini, F., Melchionna, M., Veneziano, A., Raia, P. & Manzi, G. (2017). The evolution of cranial base and face in Cercopithecoidea and Hominoidea: Modularity and morphological integration. American journal of primatology, 79(12).

`DataOrnithodirans` *Example dataset*

Description

Ornithodirans' body mass, phylogenetic tree and locomotory type (*Castiglione et al 2017*).

Usage

```
data(DataOrnithodirans)
```

Format

A list containing:

treedino Ornithodirans phylogenetic tree.

massdino numeric vector of ornithodirans body masses.

statedino vector of ornithodirans locomotory type.

Author(s)

Pasquale Raia, Silvia Castiglione, Carmela Serio, Alessandro Mondanaro, Marina Melchionna, Mirko Di Febbraro, Antonio Profico, Francesco Carotenuto

References

Castiglione, S., Tesone, G., Piccolo, M., Melchionna, M., Mondanaro, A., Serio, C., Di Febbraro Mirko, & Raia, P. A new method for testing evolutionary rate variation and shifts in phenotypic evolution. Methods in Ecology and Evolution.doi:10.1111/2041-210X.12954

evo.dir *Phylogenetic vector analysis of phenotypic change*

Description

This function quantifies direction, size and rate of evolutionary change of multivariate traits along node-to-tip paths and between species.

Usage

```
evo.dir(RR,angle.dimension=c("rates","phenotypes"),
y.type=c("original","RR"),y=NULL,pair.type=c("node","tips"),pair=NULL,
node=NULL,random=c("yes","no"),nrep=100)
```

Arguments

RR	an object produced by RRphylo.
angle.dimension	specifies whether vectors of "rates" or "phenotypes" are used.
y.type	must be indicated when angle.dimension = "phenotypes". If "original", it uses the phenotypes as provided by the user, if "RR" it uses RR$predicted.phenotypes.
y	specifies the phenotypes to be provided if y.type = "original".
pair.type	either "node" or "tips". Angles are computed between each possible couple of species descending from a specified node ("node"), or between a given couple of species ("tips").
pair	species pair to be specified if pair.type = "tips". It needs to be written as in the example below.
node	node number to be specified if pair.type = "node". Notice the node number must refer to the dichotomic version of the original tree, as produced by RRphylo.
random	whether to perform randomization test ("yes"/"no").
nrep	number of replications must be indicated if random = "yes". It is set at 100 by default.

Details

The way evo.dir computes vectors depends on whether phenotypes or rates are used as variables. RRphylo rates along a path are aligned along a chain of ancestor/descendant relationships. As such, each rate vector origin coincides to the tip of its ancestor, and the resultant of the path is given by vector addition. In contrast, phenotypic vectors are computed with reference to a common origin (i.e. the consensus shape in a geometric morphometrics). In this latter case, vector subtraction (rather than addition) will define the resultant of the evolutionary direction. It is important to realize that resultants could be at any angle even if the species (the terminal vectors) have similar phenotypes, because path resultants, rather than individual phenotypes, are being contrasted. However, the function also provides the angle between individual phenotypes as 'angle.between.species'. To

perform randomization test (`random = "yes"`), the evolutionary directions of the two species are collapsed together. Then, for each variable, the median is found, and random paths of the same size as the original paths are produced sampling at random from the 47.5th to the 52.5th percentile around the medians. This way, a random distribution of angles is obtained under the hypothesis that the two directions are actually parallel. The 'angle.direction' represents the angle formed by the species phenotype and a vector of 1s (as long as the number of variables representing the phenotype). This way, each species phenotype is contrasted to the same vector. The 'angle.direction' values could be inspected to test whether individual species phenotypes evolve towards similar directions.

Value

Under all specs, `evo.dir` returns a 'list' object. The length of the list is one if `pair.type = "tips"`. If `pair.type = "node"`, the list is as long as the number of all possible species pairs descending from the node. Each element of the list contains:

angle.path.A angle of the resultant vector of species A to MRCA

vector.size.species.A size of the resultant vector of species A to MRCA

angle.path.B angle of the resultant vector of species B to MRCA

vector.size.species.B size of the resultant vector of species B to MRCA

angle.between.species.to.mrca angle between the species paths resultant vectors to the MRCA

angle.between.species angle between species vectors (as they are, without computing the path)

MRCA the node identifying the most recent common ancestor of A and B

angle.direction.A angle of the vector of species A (as it is, without computing the path) to a fixed reference vector (the same for all species)

vec.size.direction.A size of the vector of species A

angle.direction.B angle of the vector of species B (as it is, without computing the path) to a fixed reference vector (the same for all species)

vec.size.direction.B size of the vector of species B

If `random = "yes"`, results also include p-values for the angles.

Author(s)

Pasquale Raia, Silvia Castiglione, Carmela Serio, Alessandro Mondanaro, Marina Melchionna, Mirko Di Febbraro, Antonio Profico, Francesco Carotenuto

Examples

```
  data("DataApes")
  DataApes$PCstage->PCstage
  DataApes$Tstage->Tstage

  RRphylo(tree=Tstage,y=PCstage)->RR
# Case 1. Without performing randomization test

 # Case 1.1 Computing angles between rate vectors
  # for each possible couple of species descending from node 72
```

```
  evo.dir(RR,angle.dimension="rates",pair.type="node",node=72 ,
  random="no")
 # for a given couple of species
  evo.dir(RR,angle.dimension="rates",pair.type="tips",
  pair= c("Sap_1","Tro_2"),random="no")

# Case 1.2 computing angles between phenotypic vectors provided by the user
 # for each possible couple of species descending from node 72
  evo.dir(RR,angle.dimension="phenotypes",y.type="original",
  y=PCstage,pair.type="node",node=72,random="no")
 # for a given couple of species
  evo.dir(RR,angle.dimension="phenotypes",y.type="original",
  y=PCstage,pair.type="tips",pair=c("Sap_1","Tro_2"),random="no")

# Case 1.3 computing angles between phenotypic vectors produced by "RRphylo"
 # for each possible couple of species descending from node 72
  evo.dir(RR,angle.dimension="phenotypes",y.type="RR",
  pair.type="node",node=72,random="no")
 # for a given couple of species
  evo.dir(RR,angle.dimension="phenotypes",y.type="RR",
  pair.type="tips",pair=c("Sap_1","Tro_2"),random="no")


# Case 2. Performing randomization test

 # Case 2.1 Computing angles between rate vectors
  # for each possible couple of species descending from node 72
  evo.dir(RR,angle.dimension="rates",pair.type="node",node=72 ,
  random="yes",nrep=100)

  # for a given couple of species
  evo.dir(RR,angle.dimension="rates",pair.type="tips",
  pair= c("Sap_1","Tro_2"),random="yes",nrep=100)

 # Case 2.2 computing angles between phenotypic vectors provided by the user
  # for each possible couple of species descending from node 72
  evo.dir(RR,angle.dimension="phenotypes",y.type="original",
  y=PCstage,pair.type="node",node=72,random="yes",nrep=100)

  # for a given couple of species
  evo.dir(RR,angle.dimension="phenotypes",y.type="original",
  y=PCstage,pair.type="tips",pair=c("Sap_1","Tro_2"),random="yes",nrep=100)

 # Case 2.3 computing angles between phenotypic vectors produced by "RRphylo"
  # for each possible couple of species descending from node 72
  evo.dir(RR,angle.dimension="phenotypes",y.type="RR",
  pair.type="node",node=72,random="yes",nrep=100)

  # for a given couple of species
  evo.dir(RR,angle.dimension="phenotypes",y.type="RR",
  pair.type="tips",pair=c("Sap_1","Tro_2"),random="yes",nrep=100)
```

getMommy *Upward tip or node to root path*

Description

This function is a wrapper around **phytools** `getDescendants` (*Revell 2012*). It returns the node path from a given node or species to the root of the phylogeny.

Usage

```
getMommy(tree,N,curr=NULL)
```

Arguments

`tree`	a phylogenetic tree. The tree needs not to be ultrametric and fully dichotomous.
`N`	the number of node or tip to perform the function on. Notice the function only works with number, not tip labels.
`curr`	has not to be provided by the user.

Details

The object `'curr'` is created inside the function in order to produce an array of nodes on the path.

Value

The function produces a vector of node numbers as integers, collated from a node or a tip towards the tree root.

Author(s)

Pasquale Raia, Silvia Castiglione, Carmela Serio, Alessandro Mondanaro, Marina Melchionna, Mirko Di Febbraro, Antonio Profico, Francesco Carotenuto

References

Revell, L. J. (2012). phytools: An R package for phylogenetic comparative biology (and other things). *Methods in Ecology and Evolution*, **3**, 217–223.doi:10.1111/j.2041-210X.2011.00169.x

Examples

```
data("DataApes")
DataApes$Tstage->Tstage

getMommy(tree=Tstage,N=12)
```

getSis *Get sister clade*

Description

The function identifies and returns the sister clade of a given node/tip.

Usage

```
getSis(tree,n,printZoom=c(FALSE,TRUE))
```

Arguments

tree	a phylogenetic tree. The tree needs not to be ultrametric and fully dichotomous.
n	number of focal node or name of focal tip.
printZoom	if TRUE the function plots the tree section of interest.

Value

The sister node number or sister tip name. In case of polytomies, the function returns a vector.

Author(s)

Pasquale Raia, Silvia Castiglione, Carmela Serio, Alessandro Mondanaro, Marina Melchionna, Mirko Di Febbraro, Antonio Profico, Francesco Carotenuto

Examples

```
data(DataOrnithodirans)
DataOrnithodirans$treedino->treedino
getSis(tree=treedino,n=678,printZoom=FALSE)
getSis(tree=treedino,n="Shenzhoupterus_chaoyangensis",printZoom=FALSE)
```

makeFossil *Make fossil species on a phylogeny*

Description

This function takes an object of class 'phylo' and randomly changes the lengths of the leaves.

Usage

```
makeFossil(tree,p=0.5,ex=0.5)
```

Arguments

tree	a phylogenetic tree. The tree needs not to be ultrametric and fully dichotomous.
p	the proportion of tips involved. By default it is half of the number of tips.
ex	the multiplying parameter to change the leaf lengths. It is set at 0.5 by default.

Value

The function produces a phylogeny having the same backbone of the original one.

Author(s)

Pasquale Raia, Silvia Castiglione, Carmela Serio, Alessandro Mondanaro, Marina Melchionna, Mirko Di Febbraro, Antonio Profico, Francesco Carotenuto

Examples

```
data("DataApes")
DataApes$Tstage->Tstage

makeFossil(tree=Tstage)
```

---

makeL *Matrix of branch lengths along root-to-tip paths*

Description

This function produces a $n * m$ matrix, where n=number of tips and m=number of branches (i.e. n + number of nodes). Each row represents the branch lengths aligned along a root-to-tip path.

Usage

```
makeL(tree)
```

Arguments

tree	a phylogenetic tree. The tree needs not to be ultrametric and fully dichotomous.

Value

The function returns a $n * m$ matrix of branch lengths for all root-to-tip paths in the tree (one per species).

Author(s)

Pasquale Raia, Silvia Castiglione, Carmela Serio, Alessandro Mondanaro, Marina Melchionna, Mirko Di Febbraro, Antonio Profico, Francesco Carotenuto

Examples

```
data("DataApes")
DataApes$Tstage->Tstage

makeL(tree=Tstage)
```

---

makeL1 *Matrix of branch lengths along a root-to-node path*

Description

This function produces a $n * n$ matrix, where n=number of internal branches. Each row represents the branch lengths aligned along a root-to-node path.

Usage

```
makeL1(tree)
```

Arguments

tree	a phylogenetic tree. The tree needs not to be ultrametric and fully dichotomous.

Value

The function returns a $n * n$ matrix of branch lengths for all root-to-node paths (one per each node of the tree).

Author(s)

Pasquale Raia, Silvia Castiglione, Carmela Serio, Alessandro Mondanaro, Marina Melchionna, Mirko Di Febbraro, Antonio Profico, Francesco Carotenuto

Examples

```
data("DataApes")
DataApes$Tstage->Tstage

makeL1(tree=Tstage)
```

plotRates *Plot RRphylo rates at a specified node*

Description

The function plotRates plots the RRphylo rates computed for a given clade as compared to the rates computed for the rest of the tree.

Usage

```
plotRates(RR,node,export.tiff = c(TRUE,FALSE),foldername)
```

Arguments

RR	an object produced by RRphylo.
node	the node subtending the clade of interest.
export.tiff	if TRUE the function save a "rate_bars.tiff" file inside the working directory.
foldername	the path of the folder where plots are to be found.

Value

The function produces two barplots. On the left side, the rates (in absolute values) computed for the focal clade (blue) are plotted against the rates of the rest of the tree (green). On the right side, the absolute rates of individual branches of the focal clade are collated in increasing rate value (blue bars), and contrasted to the average rate computed over the rest of the tree branches (the vertical red line). It also returns the rate values for both nodes and species descending from the focal node.

Examples

```
data("DataApes")
DataApes$PCstage->PCstage
DataApes$Tstage->Tstage

RRphylo(tree=Tstage,y=PCstage)->RR

plotRates(RR,node=72,foldername=tempdir(),export.tiff = TRUE)
```

RBC *Rate By Clade*

Description

The 'RBC' function (*Piras et al. 2018*) provides a test for Brownian rate variation via the non-censored approach (*O'Meara et al. 2006*).

Usage

```
RBC(RR,y,n.shift=c("clust","fix"),NS=3,clus=.5,f=NULL,foldername)
```

Arguments

RR	an object fitted by the function RRphylo.
y	either a single vector variable or a multivariate dataset of class 'matrix'.
n.shift	it specifies whether the function automatically searches for shifts ("clust") or the number of shifts to search for has to be set by the user ("fix").
NS	the number of shifts to be specified for n.shift = "fix". It is set at 3 by default.
clus	the proportion of clusters to be used in parallel computing.
f	the size of the smallest clade to be tested in 'RBC'. By default, nodes subtending to at least one tenth of the tree tips are tested.
foldername	the path of the folder where plots are to be found.

Details

The user indicates the minimum size of the clades to be tested for rate variation must be (by setting the argument 'f', which is by default set at one-tenth of the tree size). Individual nodes are arranged according to their rates (i.e. in descending Brownian rate value). Then, the user is left with two different options to locate the number of potential shifts. First (n.shift = "fix"), it is possible to specify the number *n* (='NS') of shifts to be searched for all combinations of the *n* nodes with the *n* largest sigma2 (Brownian rate) value, with size 1 to *n*. For instance, with 'NS' = 3 RRphylo will search through all the eight possible combinations of the 3 nodes with the largest sigma2 values (three combinations with one shift only, one for each node; three combinations of two shifts at two different nodes; and a single combination including all the three shifts for all 'NS'=3 nodes, plus Brownian motion, which means no shift applied). Alternatively (n.shift = "clust"), all nodes subtending a number of tips as large as f are partitioned in groups according to their patristic distance, and the number of distinct groups with potential shifts is established via bootstrapped cluster analysis of the internodes distances, by using **pvclust** package in R (*Suzuki and Shimodaira 2015*). This way the number of potential shifts are located in topologically distinct parts of the tree. The resulting number of groups *k* is thus taken to be equivalent to the number of shifts to be searched, by examining all possible combinations of the *k* nodes with the largest simga2 values. Of course, it is still possible (and in fact tested) that more than one shift falls in the same region of the tree. Once potential shifts are located, their combinations represent different rate variation

models, which are compared to each other (and to a single rate, Brownian motion model) by means of restricted maximum likelihood fitted with the function `brownieREML` in **phytools** (*Revell 2012*), or `mvBM` in **mvMORPH** (*Clavel et al. 2015*) in the multivariate case. The likelihoods of individual models are contrasted to each other to find the best model by means of likelihood ratio test.

Value

rbc brownian rate values for all possible clades identified (i.e. subtrees having number of tips > `f`).

best.model best Brownian rate variation ("non-censored") model selected by means of likelihood ratio test.

alternative.models alternative Brownian rate variation ("non-censored") model selected by means of LRT.

n.shifts the number of shifts for the `RBC` analysis.

Author(s)

Pasquale Raia, Silvia Castiglione, Carmela Serio, Alessandro Mondanaro, Marina Melchionna, Mirko Di Febbraro, Antonio Profico, Francesco Carotenuto

References

Piras, P., Silvestro, D., Carotenuto, F., Castiglione, S., Kotsakis, A., Maiorino, L., Melchionna, M., Mondanaro, A., Sansalone, G., Serio, C., Vero, V.A., & Raia, P. (2018). Evolution of the sabertooth mandible: A deadly ecomorphological specialization. *Palaeogeography, Palaeoclimatology, Palaeoecology*, in press

O'Meara, B. C., Ané, C., Sanderson, M. J., & Wainwright, P. C. (2006). Testing for different rates of continuous trait evolution using likelihood. *Evolution*, **60**, 922–933.

Suzuki, R., & Shimodaira, H. (2015). pvclust: Hierarchical Clustering with P-Values via Multiscale Bootstrap Resampling. R package version 2.0-0. https://CRAN.R-project.org/package=pvclust

Revell, L. J. (2012). phytools: An R package for phylogenetic comparative biology (and other things). *Methods in Ecology and Evolution*, **3**, 217–223.doi:10.1111/j.2041-210X.2011.00169.x

Clavel, J., Escarguel, G., & Merceron, G.(2015). mvMORPH: an R package for fitting multivariate evolutionary models to morphometric data. *Methods in Ecology and Evolution*, **6**:1311–1319. doi: 10.1111/2041-210X.12420

Examples

```
data("DataOrnithodirans")
DataOrnithodirans$treedino->treedino
DataOrnithodirans$massdino->massdino



    RRphylo(tree=treedino,y=massdino)->RR

# Case 1. RBC fixing the number of shifts at 2
    RBC(RR=RR,y=massdino,n.shift="fix",NS=2,foldername=tempdir())
```

```
# Case 2. RBC automatically searching for shifts
    RBC(RR=RR,y=massdino,n.shift="clust",foldername=tempdir())
```

`retrieve.angles` *Extracting a user-specified subset of the evo.dir results*

Description

This function takes the result list produced by `evo.dir` as the input, and extracts a specific subset of it. The user may specify whether to extract the set of angles between species resultant vectors and the MRCA, the size of resultant vectors, or the set of angles between species.

Usage

```
retrieve.angles(angles.res,wishlist=c("anglesMRCA","angleDir","angles.between.species"),
random=c("yes","no"),focus=c("node","species","both","none"),
node=NULL,species=NULL,write2csv=c("no","yes"))
```

Arguments

`angles.res`	the object resulting from `evo.dir` function.
`wishlist`	specifies whether to extract angles and sizes (`"anglesMRCA"`) of resultant vectors between individual species and the MRCA, angles and sizes (`"angleDir"`) of vectors between individual species and a fixed reference vector (the same for all species), or angles between species resultant vectors (`"angles.between.species"`).
`random`	it needs to be `"yes"` if `'angles.res'` object contains randomization results.
`focus`	it can be `"node"`, `"species"`, `"both"`, or `"none"`, whether the user wants the results for a focal node, or for a given species, for both, or just wants to visualize everything.
`node`	must be indicated if `focus= "node"` or `"both"`. As for `evo.dir`, the node number must refer to the dichotomic version of the original tree, as produced by `RRphylo`.
`species`	must be indicated if `focus= "species"` or `"both"`.
`write2csv`	if `"yes"` results are saved to a .csv file in your working directory.

Details

`retrieve.angles` allows to focalize the extraction to a particular node, species, or both. Otherwise it returns the whole dataset.

Value

`retrieve.angles` outputs an object of class `'data.frame'`.

If `wishlist = "anglesMRCA"`, the data frame includes:

- **MRCA** the most recent common ancestor the angle is computed to
- **species** species ID
- **angle** the angle between the resultant vector of species and the MRCA
- **vector.size** the size of the resultant vector computed from species to MRCA

If `wishlist = "angleDir"`, the data frame includes:

- **MRCA** the most recent common ancestor the vector is computed to
- **species** species ID
- **angle.direction** the angle between the vector of species and a fixed reference
- **vector.size** the size of the vector of species

If `wishlist = "angles.between.species"`, the data frame includes:

- **MRCA** the most recent common ancestor the vector is computed from
- **species** pair IDs of the species pair the "angle between species" is computed for
- **angleBTWspecies2MRCA** angle between species resultant vectors to MRCA
- **anglesBTWspecies** angle between species resultant vectors

Author(s)

Pasquale Raia, Silvia Castiglione, Carmela Serio, Alessandro Mondanaro, Marina Melchionna, Mirko Di Febbraro, Antonio Profico, Francesco Carotenuto

Examples

```
    data("DataApes")
    DataApes$PCstage->PCstage
    DataApes$Tstage->Tstage

    RRphylo(tree=Tstage,y=PCstage)->RR

# Case 1. "evo.dir" without performing randomization
    evo.dir(RR,angle.dimension="rates",pair.type="node",
    node=  57,random="no")->evo.p

 # Case 1.1 angles and sizes of resultant vectors between individual species and the MRCA:
  # for a focal node
    retrieve.angles(evo.p,wishlist="anglesMRCA",random="no",focus="node",
    node=68,write2csv="no")
  # for a focal species
    retrieve.angles(evo.p,wishlist="anglesMRCA",random="no",focus="species",
    species="Sap", write2csv="no")
  # for both focal node and species
    retrieve.angles(evo.p,wishlist="anglesMRCA",random="no",focus="both",
```

```
    node=68,species="Sap",write2csv="no")
  # without any specific requirement
    retrieve.angles(evo.p,wishlist="anglesMRCA",random="no",focus="none",
    write2csv="no")

 # Case 1.2 angles and sizes of vectors between individual species
 #and a fixed reference vector:
  # for a focal node
    retrieve.angles(evo.p,wishlist="angleDir",random="no",focus="node",
    node=68,write2csv="no")
  # for a focal species
    retrieve.angles(evo.p,wishlist="angleDir",random="no",focus="species",
    species="Sap", write2csv="no")
  # for both focal node and species
    retrieve.angles(evo.p,wishlist="angleDir",random="no",focus="both",
    node=68,species="Sap",write2csv="no")
  # without any specific requirement
    retrieve.angles(evo.p,wishlist="angleDir",random="no",focus="none",
    write2csv="no")

 # Case 1.3 angles between species resultant vectors:
  # for a focal node
    retrieve.angles(evo.p,wishlist="angles.between.species",random="no",
    focus="node", node=68,write2csv="no")
  # for a focal species
    retrieve.angles(evo.p,wishlist="angles.between.species",random="no",
    focus="species", species="Sap", write2csv="no")
  # for both focal node and species
    retrieve.angles(evo.p,wishlist="angles.between.species",random="no",
    focus="both",node=68,species="Sap",write2csv="no")
  # without any specific requirement
    retrieve.angles(evo.p,wishlist="angles.between.species",random="no",
    focus="none",write2csv="no")



# Case 2. "evo.dir" with performing randomization
    evo.dir(RR,angle.dimension="rates",pair.type="node",node=57,
    random="yes")->evo.p

 # Case 2.1 angles and sizes of resultant vectors between individual species and the MRCA:
  # for a focal node
    retrieve.angles(evo.p,wishlist="anglesMRCA",random="yes",focus="node",
    node=68,write2csv="no")
  # for a focal species
    retrieve.angles(evo.p,wishlist="anglesMRCA",random="yes", focus="species",
    species="Sap", write2csv="no")
  # for both focal node and species
    retrieve.angles(evo.p,wishlist="anglesMRCA",random="yes",focus="both",
    node=68,species="Sap",write2csv="no")
  # without any specific requirement
    retrieve.angles(evo.p,wishlist="anglesMRCA",random="yes",focus="none",
    write2csv="no")
```

```
# Case 2.2 angles and sizes of vectors between individual species and a fixed reference vector:
 # for a focal node
   retrieve.angles(evo.p,wishlist="angleDir",random="yes",focus="node",
   node=68,write2csv="no")
 # for a focal species
   retrieve.angles(evo.p,wishlist="angleDir",random="yes",focus="species",
   species="Sap", write2csv="no")
 # for both focal node and species
   retrieve.angles(evo.p,wishlist="angleDir",random="yes",focus="both",
   node=68, species="Sap",write2csv="no")
 # without any specific requirement
   retrieve.angles(evo.p,wishlist="angleDir",random="yes",focus="none",
   write2csv="no")

# Case 2.3 retrieve angles between species resultant vectors:
 # for a focal node
   retrieve.angles(evo.p,wishlist="angles.between.species",random="yes",
   focus="node", node=68,write2csv="no")
 # for a focal species
   retrieve.angles(evo.p,wishlist="angles.between.species",random="yes",
   focus="species", species="Sap", write2csv="no")
 # for both focal node and species
   retrieve.angles(evo.p,wishlist="angles.between.species",random="yes",
   focus="both",node=68,species="Sap",write2csv="no")
 # without any specific requirement
   retrieve.angles(evo.p,wishlist="angles.between.species",random="yes",
   focus="none",write2csv="no")
```

`RRphylo` *Evolutionary rates computation along phylogenies*

Description

The function `RRphylo` (*Castiglione et al. 2018*) performs the phylogenetic ridge regression. It takes a tree and a vector of tip data (phenotypes) as entries, calculates the regularization factor, produces the matrices of tip to root (`makeL`), and node to root distances (`makeL1`), the vector of ancestral state estimates, and the rates vector for all the branches of the tree. For multivariate data, rates are given as both one vector per variable, and as a multivariate vector obtained by computing the Euclidean Norm of individual rate vectors.

Usage

```
RRphylo(tree,y,cov=NULL)
```

Arguments

`tree`	a phylogenetic tree. The tree needs not to be ultrametric or fully dichotomous.

y	either a single vector variable or a multivariate dataset of class 'matrix'.
cov	the covariate to be indicated if its effect on the rates must be accounted for. In this case residuals of the covariate versus the rates are used as rates. 'cov' must be as long as the number of nodes plus the number of tips of the tree, which can be obtained by running RRphylo on the covariate as well, and taking the vector of ancestral states and tip values to form the covariate, as in the example below.

Value

tree the tree used by RRphylo. The fully dichotomous version of the tree argument. For trees with polytomies, the tree is resolved by using multi2di function in the package **ape**. If the latter is a dichotomous tree, the two trees will be identical.

tip.path a $n * m$ matrix, where n=number of tips and m=number of branches (i.e. 2*n-1). Each row represents the branch lengths along a root-to-tip path.

node.path a $n * n$ matrix, where n=number of internal branches. Each row represents the branch lengths along a root-to-node path.

rates single rate values computed for each branch of the tree. If y is a single vector variable, rates are equal to multiple.rates. If y is a multivariate dataset, rates are computed as the square root of the sum of squares of each row of $multiple.rates.

aces the phenotypes reconstructed at nodes.

predicted.phenotypes the vector of estimated tip values.

multiple.rates a $n * m$ matrix, where n= number of branches (i.e. n*2-1) and m = number of variables. For each branch, the column entries represent the evolutionary rate.

lambda the regularization factor fitted within RRphylo inner function optL.

Author(s)

Pasquale Raia, Silvia Castiglione, Carmela Serio, Alessandro Mondanaro, Marina Melchionna, Mirko Di Febbraro, Antonio Profico, Francesco Carotenuto

References

Castiglione, S., Tesone, G., Piccolo, M., Melchionna, M., Mondanaro, A., Serio, C., Di Febbraro, M., & Raia, P.(2018). A new method for testing evolutionary rate variation and shifts in phenotypic evolution. *Methods in Ecology and Evolution*, in press.doi:10.1111/2041-210X.12954

Examples

```
data("DataOrnithodirans")
DataOrnithodirans$treedino->treedino
DataOrnithodirans$massdino->massdino

# Case 1. "RRphylo" without accounting for the effect of a covariate
    RRphylo(tree=treedino,y=massdino)

# Case 2. "RRphylo" accounting for the effect of a covariate
  # "RRphylo" on the covariate in order to retrieve ancestral state values
```

```
RRphylo(tree=treedino,y=massdino)->RRcova
c(RRcova$aces,massdino)->cov.values
c(rownames(RRcova$aces),names(massdino))->names(cov.values)

RRphylo(tree=treedino,y=massdino,cov=cov.values)
```

---

search.shift *Locating shifts in phenotypic evolutionary rates*

Description

The function search.shift (*Castiglione et al. 2018*) tests whether individual clades or isolated tips dispersed through the phylogeny evolve at different RRphylo rates as compared to the rest of the tree. Instances of rate shifts may be automatically found.

Usage

```
search.shift(RR, status.type = c("clade", "sparse"),
auto.recognize = c("yes","no"), node = NULL, test.single = c("yes", "no"),
state = NULL, cov = NULL, nrep = 1000, f = NULL,foldername)
```

Arguments

RR	an object fitted by the function RRphylo.
status.type	whether the "clade" or "sparse" condition must be tested.
auto.recognize	indicates whether individual clades must be tested for rate shift without any a priori indication of particular nodes to be tested.
node	under the "clade" condition, the node of the tree to be tested for the rate shift. When multiple nodes are tested, they need to be written as in the example below.
test.single	whether multiple nodes indicated under the "clade" condition should be tested individually ("yes") or not ("no").
state	the state of the tips specified under the "sparse" condition.
cov	the covariate to be indicated if its effect on rate values must be accounted for. Contrary to RRphylo, 'cov' needs to be as long as the number of tips of the tree.
nrep	the number of simulations to be performed for the rate shift test, by default nrep is set at 1000.
f	the size of the smallest clade to be tested. By default, nodes subtending to one tenth of the tree tips are tested.
foldername	the path of the folder where plots are to be found.

Details

The function search.shift takes the object produced by RRphylo. Two different conditions of rate change can be investigated. Under the "clade" condition the vector of node or nodes subjected to the shift must be provided. Alternatively, under the "sparse" case the (named) vector of states (indicating which tips are or are not evolving under the rate shift according to the tested hypothesis) must be indicated. In the "clade" case, the function may perform an 'auto.recognize' feature. Under such specification, the function automatically tests individual clades (from clades as large as one half of the tree down to a specified clade size) for deviation of their rates from the background rate of the rest of the tree, which is identical to the "clade" case. An inclusive clade with significantly high rates is likely to include descending clades with similarly significantly high rates. Hence, with 'auto.recognize' the search.shift function is written as to scan clades individually and to select only the node subtending to the highest difference in mean absolute rates as compared to the rest of the tree. A plot of the tree highlighting nodes subtending to significantly different rates is automatically produced. Caution must be put into interpreting the 'auto.recognize' results. For instance, a clade with small rates and another with large rates could be individuated even under BM. This does not mean these clades are actual instances for rate shifts. Clades must be tested on their own without the 'auto.recognize' feature, which serves as guidance to the investigator, when no strong a priori hypothesis to be tested is advanced. The 'auto.recognize' feature is not meant to provide a test for a specific hypothesis. It serves as an optional guidance to understand whether and which clades show significantly large or small rates as compared to the rest of the tree. Individual clades are tested at once, meaning that significant instances of rate variation elsewhere on the tree are ignored. Conversely, running the "clade" condition without including the 'auto.recognize' feature, multiple clades presumed to evolve under the same shift are tested together, meaning that their rates are collectively contrasted to the rest of the tree, albeit they can additionally be compared to each other upon request. Under both the "clade" and "sparse" conditions, multiple clades could be specified at once, and optionally tested individually (for deviation of rates) against the rates of the rest of the tree and against each other. The histogram of random differences of mean rates distribution along with the position of the real difference of means is automatically generated by search.shift. Regardless of which condition is specified, the function output produces the real difference of means, and their significance value.

Value

Under all circumstances, search.shift provides a vector of $rates. If 'cov' values are provided, rates are regressed against the covariate and the residuals of such regression form the vector $rates. Otherwise, $rates are the same rates as with the RR argument.

Under 'auto.recognize', a list including:

$'all different clades' the probability that the nodes subtending to the largest and smallest average rates (in absolute values) do represent a real shift. Probabilities are contrasted to simulations shuffling the rates across the tree branches, a number of times specified by the argument nrep. Note that the p-values refer to the number of times the real average rates are larger (or smaller) than the rates averaged over the rest of the tree, divided by the number of simulations. Hence, large rates are significantly larger than the rest of the tree (at alpha = 0.05), when the probability is > 0.975; and small rates are significantly small for $p < 0.025$.

$'all clades rate difference' the nodes subtending to the largest and smallest rates (in absolute values) in the tree, and the average rate differences, computed as the mean rate over all branches subtended by the node, minus the average rate for the rest of the tree.

$'shift test single clades' the same as with 'all different clades' but restricted to the largest/smallest rate values along a single clade (i.e. nested clades with smaller rate shifts are excluded). Large rates are significantly larger than the rest of the tree (at alpha = 0.05), when the probability is > 0.975; and small rates are significantly small for p < 0.025.

$'average rate difference' the average rate differences for the branches descending by the nodes indicated in 'shift test single clades'.

`'clade'` condition, without `'auto.recognize'`:

$'shift test' this specifies the significance of the rate shift test, by considering all the specified nodes as a evolving under a single rate. As with the `'auto.recognize'` feature, large rates are significantly larger than the rest of the tree (at alpha = 0.05), when the probability is > 0.975; and small rates are significantly small for p < 0.025.

$'shift test single clades' if `test.single = "yes"`, this gives the significance for individual clades, tested separately. As with the `'auto.recognize'` feature, large rates are significantly larger than the rest of the tree (at alpha = 0.05), when the probability is > 0.975; and small rates are significantly small for p < 0.025.

$'average rate difference' this is the average rate difference, computed as the mean rate over all branches subtended by the node, minus the average rate for the rest of the tree. Nodes along a single path are not permitted. In this case, only the largest/smallest rate values are selected large rates are significantly larger than the rest of the tree (at alpha = 0.05), when the probability is > 0.975; and small rates are significantly small for p < 0.025.

Under the `'sparse'` condition:

$'rate difference' this is the average rate difference, computed as the mean rate over all leaves (terminal nodes) evolving under a given state, minus the average rate for each other state.

$'p rate diff' the probability that the shift under the sparse condition is real. Large rates are significantly larger than the rest of the tree (at alpha = 0.05), when the probability is > 0.975; and small rates are significantly small for p < 0.025. States are compare pairwise.

Author(s)

Pasquale Raia, Silvia Castiglione, Carmela Serio, Alessandro Mondanaro, Marina Melchionna, Mirko Di Febbraro, Antonio Profico, Francesco Carotenuto

References

Castiglione, S., Tesone, G., Piccolo, M., Melchionna, M., Mondanaro, A., Serio, C., Di Febbraro, M., & Raia, P.(2018). A new method for testing evolutionary rate variation and shifts in phenotypic evolution. *Methods in Ecology and Evolution*, in press.doi:10.1111/2041-210X.12954

Examples

```
data("DataOrnithodirans")
DataOrnithodirans$treedino->treedino
DataOrnithodirans$massdino->massdino
DataOrnithodirans$statedino->statedino


    RRphylo(tree=treedino,y=massdino)->dinoRates
```

```
# Case 1. Without accounting for the effect of a covariate

 # Case 1.1 "clade" condition
  # with auto.recognize
    search.shift(dinoRates,auto.recognize="yes",test.single= "no",
    status.type= "clade",foldername=tempdir())
  # testing two hypothetical clades
    search.shift(dinoRates,auto.recognize= "no",test.single= "yes",
    status.type= "clade", node=c(697,746),foldername=tempdir())

 # Case 1.2 "sparse" condition
  # testing the sparse condition.
    search.shift(dinoRates,auto.recognize= "no",test.single= "no",
    status.type= "sparse", state=statedino,foldername=tempdir())


# Case 2. Accounting for the effect of a covariate

 # Case 2.1 "clade" condition
    search.shift(dinoRates,auto.recognize= "yes",test.single= "no",
    status.type= "clade", cov=massdino,foldername=tempdir())

 # Case 2.2 "sparse" condition
   search.shift(dinoRates,status.type="sparse",state=statedino,cov=massdino,foldername=tempdir())
```

---

setBM *Producing simulated phenotypes with trends*

Description

The function setBM is wrapper around **phytools** fastBM function, which generates BM simulated phenotypes with or without a trend.

Usage

```
setBM(tree, nY = 1, s2 = 1, a = 0, type = c("", "brown","trend", "drift"),
tr = 10, t.shift = 0.5, trend.type = c("linear", "stepwise"))
```

Arguments

tree	a phylogenetic tree.
nY	the number of phenotypes to simulate.
s2	value of the Browian rate to use in the simulations.
a	the phenotype at the tree root.

type	the type of phenotype to simulate. With the option "brown" the phenotype will have no trend in the mean or in the rate of evolution (actually the residuals of the phenotype versus time regression, established via Breusch-Pagan test). A variation in the phenotypic mean over time (a phenotypic trend) is obtained by selection the option "drift". A trend in the rate of evolution should produce an increased variance in the residuals over time, which can be inspected under the option "trend", by means of Breusch-Pagan test for residuals heteroscedasticity.
tr	the intensity of the trend in homoscedasticity with the "stepwise" option is controlled by the 'tr' argument. The scalar 'tr' is the multiplier of the branches extending after the shift point as indicated by 't.shift'.
t.shift	the relative time distance from the tree root where the stepwise increase in the rate of evolution is indicated to apply.
trend.type	two kinds of heteroscedastic residuals are generated under the "trend" type. The option "linear" produces a linear increase in heteroscedasticity, whereas the "stepwise" option produces an increase after a specified point in time.

Details

Note that setBM differs from fastBM in that the produced phenotypes are checked for the existence of a temporal trend either in the phenotypes themselves, or in the residuals of the phenotype versus age regression. The user may specify whether she wants trendless data (option "brown"), phenotypes trending in time (option "drift"), or phenotypes whose variance increases over time, consistently with the possible existence of a trend in the rate of evolution (option "trend"). In the latter case, the user may indicate the intensity of the change in variance, and whether it should occur after a given proportion of the tree height (hence a given point back in time). Trees in setBM are treated as non ultrametric. If an ultrametric tree is fed to the function, setBM alters slightly the leaf lengths multiplying randomly half of the leaves by 1 * 10-3.

Value

Either an object of class 'array' containing a single phenotype or an object of class 'matrix' of *n* phenotypes as columns, where *n* is indicated as nY = *n*.

Author(s)

Pasquale Raia, Silvia Castiglione, Carmela Serio, Alessandro Mondanaro, Marina Melchionna, Mirko Di Febbraro, Antonio Profico, Francesco Carotenuto

Examples

```
data("DataOrnithodirans")
DataOrnithodirans$treedino->treedino

setBM(tree=treedino, nY= 1, type="brown")
setBM(tree=treedino, nY= 1, type="drift")
setBM(tree=treedino, nY= 1, type="trend", trend.type="linear")
```

`sizedsubtree` *Find a node subtending to a clade of desidered size*

Description

The function `sizedsubtree` scans a phylogentic tree to randomly find nodes subtending to a subtree of desired minimum size, up to one half of the tree size (number of tips).

Usage

```
sizedsubtree(tree,Size=NULL,time.limit=10)
```

Arguments

`tree`	a phylogenetic tree.
`Size`	the desired size of the tree subtending to the extracted node. By default, the minimum tree size is set at one tenth of the tree size (i.e. number of tips).
`time.limit`	specifies a limit to the searching time, a warning message is thrown if the limit is reached.

Details

The argument `time.limit` sets the searching time. The algorithm stops if that limit is reached, avoiding recursive search when no solution is in fact possible.

Value

A node subtending to a subtree of desired minimum size.

Author(s)

Pasquale Raia, Silvia Castiglione, Carmela Serio, Alessandro Mondanaro, Marina Melchionna, Mirko Di Febbraro, Antonio Profico, Francesco Carotenuto

Examples

```
data("DataOrnithodirans")
DataOrnithodirans$treedino->treedino

sizedsubtree(tree=treedino,Size=40)
```

`swap.phylo` *Test the effect of phylogenetic uncertainty on rate shifts found at a particular node*

Description

The function uses a number of alternative phylogenies with altered (as compared to the reference tree) topology and branch lengths tests whether the tips descending from the specified node (`'node'`) have statistically different rates from the rest of the tree. A phenotypic vector `'y'` must be supplied. Eventually, the effect of a covariate could be included.

Usage

```
swap.phylo(tree,si=0.5,si2=0.5,node=NULL,y=NULL,rts,nrep=100,cov=NULL)
```

Arguments

`tree`	a phylogenetic tree. The tree needs not to be ultrametric or fully dichotomous.
`si`	the proportion of tips whose topologic arrangement will be swapped.
`si2`	the proportion of nodes whose age will be changed.
`node`	the focal node to be tested.
`y`	the phenotype under testing.
`rts`	the rates found by `RRphylo` on the original tree.
`nrep`	the number of simulated trees to be produced.
`cov`	the covariate to be indicated if its effect on rate values must be accounted for. Contrary to `RRphylo`, `'cov'` needs to be as long as the number of tips of the tree.

Details

`swap.phylo` changes the tree topology and branch lengths to a level specified by the user. Up to half of the tips, and half of the branch lengths can be changed randomly. The function provides a 'swapped' tree, yet, importantly, once a shift in the rate of evolution has been found by `RRphylo`, this function can be used to test whether the shift depends on the tree topology and branch lengths. It runs `RRphylo` on swapped trees (default is 100) and then calculates the absolute rate difference between all the branches of the shifted node and the rest of the tree. A t-test is eventually performed to assess significance.

Value

The function returns a 'list' object containing:

$p.swap the probability that the rates at `'node'` are different from rates at the rest of the tree.

$rates the distribution of rates per branch as calculated by `RRphylo` on 'swapped' phylogenies.

Examples

```
data("DataApes")
DataApes$PCstage->PCstage
DataApes$Tstage->Tstage
DataApes$CentroidSize->CS


# Case 1. swap.phylo without accounting for the effect of a covariate
RRphylo(tree=Tstage,y=PCstage)->RR
RR$rates->rr
swap.phylo(Tstage,node=61,y=PCstage,rts=rr)
# Case 2. swap.phylo accounting for the effect of a covariate
RRphylo(tree=Tstage,y=CS)->RRcova
c(RRcova$aces,CS)->cov.values
c(rownames(RRcova$aces),names(CS))->names(cov.values)
RRphylo(tree=Tstage,y=PCstage,cov=cov.values)->RR
RR$rates->rr
swap.phylo(Tstage,node=61,y=PCstage,rts=rr,cov=CS)
```

---

swapONE *Create alternative phylogenies from a given tree*

Description

The function produces an alternative phylogeny with altered topology and branch length, and computes the Kuhner-Felsenstein (Kuhner & Felsenstein 1994) distance between original and 'swapped' tree.

Usage

```
swapONE(tree,si=0.5,si2=0.5)
```

Arguments

tree	a phylogenetic tree. The tree needs not to be ultrametric or fully dichotomous.
si	the proportion of tips whose topologic arrangement will be swapped.
si2	the proportion of nodes whose age will be changed.

Details

swap.one changes the tree topology and branch lengths. Up to half of the tips, and half of the branch lengths can be changed randomly. Each randomly selected node is allowed to move up to 2 nodes apart from its original position.

Value

The function returns a list containing the 'swapped' version of the original tree, and the Kuhner-Felsenstein distance between the trees.

Author(s)

Pasquale Raia, Silvia Castiglione, Carmela Serio, Alessandro Mondanaro, Marina Melchionna, Mirko Di Febbraro, Antonio Profico, Francesco Carotenuto

References

Kuhner, M. K. & Felsenstein, J. (1994). A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates, *Molecular Biology and Evolution*, **11**, 459-–468

Examples

```
data("DataOrnithodirans")
DataOrnithodirans$treedino->treedino

swapONE(tree=treedino)
```

Index