Package 'RRphylo'

August 29, 2022

Type Package

Title Phylogenetic Ridge Regression Methods for Comparative Studies

Date 2022-08-19

Version 2.7.0

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., 2018 <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, the computation of ontogenetic shape vectors and test for morphological convergence.

License GPL-2

Encoding UTF-8

LazyData true

Depends R (>= 3.6.0), emmeans(>= 1.4.3)

Imports ape, phytools, stats4, foreach, doParallel, parallel

Suggests phangorn, rlist, scales, R.utils,cluster, RColorBrewer, nlme, car, smatr, picante, vegan,ddpcr, geomorph, rmarkdown, knitr,kableExtra, plotrix, pdftools, rgl, mvMORPH, ggplot2, qpdf, inflection, Rvcg, Morpho, evolqg, manipulate, markdown, Rphylopars, phylolm, webshot2

RoxygenNote 7.2.1

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2022-08-29 12:40:02 UTC

1

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 phenotypic 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). Citations to individual functions are available by typing citation("RRphylo").

Author(s)

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

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,clus=0.5)
```

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".

clus   the proportion of clusters to be used in parallel computing. To run the single-
                    threaded version of angle.matrix set clus = 0.

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. Secondly, it
specifies whether ontogenetic vectors to MRCA, and intraspecific ontogenetic vectors significantly
differ in angle or size between species pairs. Then, for each species pair, it indicates if the biogenetic
law and paedomorphosis apply.

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

```
## Not run:
data("DataApes")
DataApes$PCstage->PCstage
DataApes$Tstage->Tstage
DataApes$CentroidSize->CS

cc<- 2/parallel::detectCores()
RRphylo(tree=Tstage,y=PCstage,clus=cc)->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",clus=cc)
  # on rates vectors
    angle.matrix(RR,node=72,Y=PCstage,select.axes="yes",type="rates",clus=cc)

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


# 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,clus=cc)
```

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

## End(Not run)
```

---

conv.map   *Mapping morphological convergence on 3D surfaces*

Description

Given vectors of RW (or PC) scores, the function selects the RW(PC) axes which best account for
convergence and maps convergent areas on the corresponding 3D surfaces.

Usage

```
conv.map(dataset,pcs,mshape,conv=NULL, exclude=NULL,out.rem=TRUE,
  show.consensus=FALSE, plot=TRUE,col="blue",names = TRUE)
```

Arguments

dataset	data frame (or matrix) with the RW (or PC) scores of the group or species to be compared.
pcs	RW (or PC) vectors (eigenvectors of the covariance matrix) of all the samples.
mshape	the Consensus configuration.
conv	a named character vector indicating convergent species as (indicated as "conv" in dataset) and not convergent species (indicated as "noconv").
exclude	integer: the index number of the RW (or PC) to be excluded from the comparison.
out.rem	logical: if TRUE triangles with outlying area difference are removed.
show.consensus	logical: if TRUE, the Consensus configuration is included in the comparison.
plot	logical: if TRUE, the pairwise comparisons are be plotted. For more than 5 pairwise comparisons, the plot is not shown.
col	character: the colour for the plot.
names	logical: if TRUE, the names of the groups or species are displayed in the 3d plot.

Details

conv.map automatically builds a 3D mesh on the mean shape calculated from the Relative Warp
Analysis (RWA) or Principal Component Analysis (PCA) (*Schlager 2017*) by applying the function
[vcgBallPivoting](#) (**Rvcg**). conv.map further gives the opportunity to exclude some RW (or PC)
axes from the analysis because, for example, in most cases the first axes are mainly related to
high-order morphological differences driven by phylogeny and size variations. conv.map finds and
plots the strength of convergence on 3D surfaces. An output of conv.map (if the dataset contains
a number equal or lower then 5 items) is an interactive plot mapping the convergence on the 3D

models. In the upper triangle of the 3D multiple layouts the rows representing the reference models and the columns the target models. On the contrary, on the lower triangle the rows correspond to the target models and the columns the reference models. In the calculation of the differences of areas we supply the possibility to find and remove outliers from the vectors of areas calculated on the reference and target surfaces. We suggest considering this possibility if the mesh may contain degenerate facets.

Value

The function returns a list including:

- **$angle.compare** data frame including the real angles between the given shape vectors, the angles conv computed between vectors of the selected RWs (or PCs), the angles between vectors of the non-selected RWs (or PCs), the difference conv, and its p values.

- **$selected.pcs** RWs (or PCs) axes selected for convergence.

- **$average.dist** symmetric matrix of pairwise distances between 3D surfaces.

- **$suface1** list of coloured surfaces, if two meshes are given, it represents convergence between mesh A and B charted on mesh A.

- **$suface2** list of coloured surfaces, if two meshes are given, it represents convergence between mesh A and B charted on mesh B.

- **$scale** the value used to set the colour gradient, computed as the maximum of all differences between each surface and the mean shape.

Author(s)

Marina Melchionna, Antonio Profico, Silvia Castiglione, Carmela Serio, Gabriele Sansalone, Pasquale Raia

References

Schlager, S. (2017). *Morpho and Rvcg–Shape Analysis in R: R-Packages for geometric morphometrics, shape analysis and surface manipulations.* In: Statistical shape and deformation analysis. Academic Press. Melchionna, M., Profico, A., Castiglione, S., Serio, C., Mondanaro, A., Modafferi, M., Tamagnini, D., Maiorano, L. , Raia, P., Witmer, L.M., Wroe, S., & Sansalone, G. (2021). A method for mapping morphological convergence on three-dimensional digital models: the case of the mammalian sabre-tooth. Palaeontology, 64, 573–584. doi:10.1111/pala.12542

See Also

search.conv vignette ; relWarps ; procSym

Examples

```
## Not run:
data(DataSimians)
DataSimians$pca->pca

## Case 1. Convergent species only
   dato<-pca$PCscores[c(1,4),]
```

```
      CM<-conv.map(dataset = dato,
                   pcs = pca$PCs,
                   mshape = pca$mshape,
                   show.consensus = TRUE)

  ## Case 2. Convergent and non-convergent species
      dato<-pca$PCscores[c(1,4,7),]
      conv<-c("conv","conv","noconv")
      names(conv)<-rownames(dato)

      CM<-conv.map(dataset = dato,
                   pcs = pca$PCs,
                   mshape = pca$mshape,
                   conv = conv,
                   show.consensus = TRUE,
                   col = "orange")

  ## End(Not run)
```

---

cutPhylo   *Cut the phylogeny at a given age or node*

Description

The function cuts all the branches of the phylogeny which are younger than a specific age or node
(i.e. the age of the node).

Usage

```
cutPhylo(tree,age=NULL,node=NULL,keep.lineage=TRUE)
```

Arguments

tree	a phylogenetic tree. The tree needs not to be ultrametric and fully dichotomous.
age	the age (in terms of time distance from the recent) at which the tree must be cut
node	the node whose age must be used as cutting limit.
keep.lineage	logical specifying whether lineages with no descendant tip must be retained (see example below). Default is TRUE.

Details

When an entire lineage is cut (i.e. one or more nodes along a path) and keep.lineages = TRUE, the
leaves left are labeled as "l" followed by a number.

Value

The function returns the cut phylogeny and plots it into the graphic device. The time axis keeps the
root age of the original tree. Note, tip labels are ordered according to their position in the tree.

Author(s)

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

See Also

[cutPhylo vignette](#)

Examples

```
## Not run:
library(ape)

set.seed(22)
rtree(100)->tree
3->age

cutPhylo(tree,age=age)->t1
cutPhylo(tree,age=age,keep.lineage=FALSE)->t1
cutPhylo(tree,node=151)->t2
cutPhylo(tree,node=151,keep.lineage=FALSE)->t2

## End(Not run)
```

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: e22721.

DataCetaceans   *Example dataset*

Description

Cetaceans' body and brain mass, and phylogenetic tree (Serio et al. 2019).

Usage

```
data(DataCetaceans)
```

Format

A list containing:

treecet Cetaceans phylogenetic tree.

masscet numeric vector of cetaceans body masses (ln g).

brainmasscet numeric vector of cetaceans brain masses (ln g).

aceMyst body mass (ln g) for *Mystacodon selenensis*, used as node prior at the ancestor of the Mysticeti.

Author(s)

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

References

Serio, C., Castiglione, S., Tesone, G., Piccolo, M., Melchionna, M., Mondanaro, A., Di Febbraro, M., & Raia, P. (2019). Macroevolution of Toothed Whales Exceptional Relative Brain Size. *Evolutionary Biology*, 46: 332-342. doi:10.1007/s11692-019-09485-7

DataFelids   *Example dataset*

Description

Geometric morphometrics shape data regarding felids' mandible and phylogentic tree (*Piras et al.,* *2018*).

Usage

```
data(DataFelids)
```

Format

A list containing:

$PCscoresfel A data frame containing 83 shape variables for felids' mandible.

$treefel Phylogenetic tree of felids.

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 saber-tooth mandible: A deadly ecomorphological specialization. *Palaeogeography, Palaeoclimatology, Palaeoecology*, 496, 166-174.

DataOrnithodirans   *Example dataset*

Description

Ornithodirans' body mass, phylogenetic tree and locomotory type (*Castiglione et al 2018*).

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, M., & Raia, P.(2018). A new method for testing evolutionary rate variation and shifts in phenotypic evolution. *Methods in Ecology and Evolution*, 9: 974-983.doi:10.1111/2041-210X.12954

DataSimians   *Example dataset*

Description

The output of Procrustes superimposition as performed by the function procSym on 9 simians faces and the phylogenetic tree for such species.

Usage

```
data(DataSimians)
```

Author(s)

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

---

DataUng   *Example dataset*

Description

Geometric morphometrics shape data regarding mandible and phylogentic tree of 'Ungulatomor-pha' (*Raia et al., 2010*).

Usage

```
data(DataUng)
```

Format

A list containing:

$PCscoresung A data frame containing 205 shape variables for mandible of 'Ungulatomorpha'.

$treeung Phylogenetic tree of 'Ungulatomorpha'.

$stateung vector of 'Ungulatomorpha' feeding type.

Author(s)

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

References

Raia, P., Carotenuto, F., Meloro, C., Piras, P., & Pushkina, D. (2010). The shape of contention: adaptation, history, and contingency in ungulate mandibles. *Evolution*, 64: 1489-1503.

---

distNodes   *Finding distance between nodes and tips*

Description

The function computes the distance between pairs of nodes, pairs of tips, or between nodes and tips. The distance is meant as both patristic distance and the number of nodes intervening between the pair.

Usage

```
distNodes(tree,node=NULL,clus=0.5)
```

Arguments

`tree`	a phylogenetic tree. The tree needs not to be ultrametric and fully dichotomous.
`node`	either a single node/tip or a pair of nodes/tips.
`clus`	the proportion of clusters to be used in parallel computing. To run the single-threaded version of `distNodes` set `clus = 0`.

Value

If `node` is specified, the function returns a data frame with distances between the focal node/tip and the other nodes/tips on the tree (or for the selected pair only). Otherwise, the function returns a matrix containing the number of nodes intervening between each pair of nodes and tips.

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

cc<- 2/parallel::detectCores()
distNodes(tree=Tstage,clus=cc)
distNodes(tree=Tstage,node=64,clus=cc)
distNodes(tree=Tstage,node="Tro_2",clus=cc)
distNodes(tree=Tstage,node=c(64,48),clus=cc)
distNodes(tree=Tstage,node=c(64,"Tro_2"),clus=cc)
```

---

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

```
## Not run:
  data("DataApes")
  DataApes$PCstage->PCstage
  DataApes$Tstage->Tstage
  cc<- 2/parallel::detectCores()

  RRphylo(tree=Tstage,y=PCstage, clus=cc)->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=10)

     # for a given couple of species
       evo.dir(RR,angle.dimension="rates",pair.type="tips",
       pair= c("Sap_1","Tro_2"),random="yes",nrep=10)

   # 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=10)

     # 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=10)

   # 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=10)

     # 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=10)

  ## End(Not run)
```

---

fix.poly   *Resolving polytomies to non-zero length branches*

Description

The function either collapses clades under a polytomy or resolves polytomous clades to non-zero
length branches, dichotomous clades.

Usage

```
fix.poly(tree,type=c("collapse","resolve"),node=NULL,tol=1e-10,random=TRUE)
```

Arguments

`tree`	a phylogenetic tree.
`type`	either 'collapse' to create polytomies to one or more specific nodes or 'resolve' to resolve (fix) all the polytomies within the tree or to one or more specific nodes.
`node`	the node in the tree where a polytomy should be created or fixed, either. If `type='resolve'` and `node=NULL` all the polytomies present in the tree are resolved.
`tol`	the tolerance to consider a branch length significantly greater than zero, set at 1e-10 by default. If `type='resolve'`, all the branch lengths smaller than `tol` are treated as polytomies.
`random`	a logical value specifying whether to resolve the polytomies randomly (the default) or in the order they appear in the tree (if `random = FALSE`).

Details

Under `type='resolve'` polytomous clades are resolved adding non-zero length branches to each new node. The evolutionary time attached to the new nodes is partitioned equally below the dichotomized clade.

Value

A phylogenetic tree with randomly fixed (i.e. `type='resolve'`) polytomies or created polytomies (i.e. `type='collapse'`).Note, tip labels are ordered according to their position in the tree.

Author(s)

Silvia Castiglione, Pasquale Raia, Carmela Serio

References

Castiglione, S., Serio, C., Piccolo, M., Mondanaro, A., Melchionna, M., Di Febbraro, M., Sansalone, G., Wroe, S., & Raia, P. (2020). The influence of domestication, insularity and sociality on the tempo and mode of brain size evolution in mammals. *Biological Journal of the Linnean Society*,132: 221-231. doi:10.1093/biolinnean/blaa186

See Also

fix.poly vignette;

Examples

```
## Not run:
 require(ape)

 data("DataCetaceans")
 DataCetaceans$treecet->treecet

 # Resolve all the polytomies within Cetaceans phylogeny
```

```
fix.poly(treecet,type="resolve")->treecet.fixed
par(mfrow=c(1,2))
plot(treecet,no.margin=TRUE,show.tip.label=FALSE)
plot(treecet.fixed,no.margin=TRUE,show.tip.label=FALSE)

# Resolve the polytomies pertaining the genus Kentriodon
fix.poly(treecet,type="resolve",node=221)->treecet.fixed2
par(mfrow=c(1,2))
plot(treecet,no.margin=TRUE,show.tip.label=FALSE)
plot(treecet.fixed2,no.margin=TRUE,show.tip.label=FALSE)

# Collapse Delphinidae into a polytomous clade
fix.poly(treecet,type="collapse",node=179)->treecet.collapsed
par(mfrow=c(1,2))
plot(treecet,no.margin=TRUE,show.tip.label=FALSE)
plot(treecet.collapsed,no.margin=TRUE,show.tip.label=FALSE)

## End(Not run)
```

---

getGenus   *Taxonomic inspection of the tree at the genus level*

Description

The function returns the most recent common ancestor and the number of species belonging to each or some user-specified genera within the phylogenetic tree.

Usage

```
getGenus(tree,genera=NULL)
```

Arguments

tree
: a phylogenetic tree. The tree needs not to be ultrametric and fully dichotomous. Generic name and specific epithet must be separated by '_'.

genera
: a character vector including one or more genera to focus on. Please notice the function is case sensitive.

Value

The function returns a data-frame including the number of species and the most recent common ancestor of each genera.

Author(s)

Silvia Castiglione, Pasquale Raia, Carmela Serio

Examples

```
DataCetaceans$treecet->tree

getGenus(tree)
getGenus(tree,c("Mesoplodon","Balaenoptera"))
```

---

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)
```

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. The function also works with tip labels.

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=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=677,printZoom=FALSE)
getSis(tree=treedino,n="Shenzhoupterus_chaoyangensis",printZoom=FALSE)
```

---

lollipoPlot   *Lollipop charts*

Description

The function generates lollipop or dumbbell dots charts.

Usage

```
lollipoPlot(values, type = "v", pt.lwd = NULL, pt.col = NULL, ...)
```

Arguments

`values`	either a vector, matrix, or data.frame of data. If matrix or data.frame including two columns, a dumbbell dots chart is plotted.
`type`	plot direction, either vertical ("v", the default) or horizontal ("h").
`pt.lwd`	points lwd
`pt.col`	points color
`...`	other arguments passed to the functions `plot`, `points`, and `segments`.

Details

If a dumbbell dots chart is plotted, different parameters (i.e. col/cex/pch/bg/lwd) for starting and ending points can be supplied. See example for further details.

Author(s)

Silvia Castiglione, Carmela Serio, Pasquale Raia

Examples

```
require(emmeans)

lollipoPlot(values=feedlot[,4],pt.col="green",pt.lwd=2,lwd=0.8,col="gray20",
            ylab="swt",xlab="samples")

line.col<-sample(colors()[-1],length(levels(feedlot[,1])))
line.col<-rep(line.col,times=table(feedlot[,1]))

lollipoPlot(values=feedlot[order(feedlot[,1]),3],ylab="ewt",xlab="samples",
            bg=as.numeric(as.factor(feedlot[order(feedlot[,1]),2])),
            cex=1.2,pch=21,col=line.col)


lollipoPlot(values=feedlot[order(feedlot[,1]),3:4],type="h",ylab="ewt",xlab="samples",
            pt.col=c("blue","cyan"),cex=1.2,pch=c(3,4),col=line.col)

lollipoPlot(values=feedlot[order(feedlot[,1]),3:4],type="h",ylab="ewt",xlab="samples",
            bg=cbind(line.col,line.col),cex=c(1.2,1),pch=c(21,22))
```

---

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)
```

---

namesCompare   *Checking species names for misspelling and synonyms*

Description

The function cross-references two vectors of species names checking for possible synonyms, misspelled names, and genus-species or species-subspecies correspondence.

Usage

```
namesCompare(vec1, vec2)
```

Arguments

vec1, vec2   a vector of species names. Genus names only are also allowed. Generic name and specific epithet must be separated by '_'. Note that vec2 is used as the reference. Incomplete or suspicious names are better placed in vec1 (see example below).

Value

The function returns a `list` including:

$genus if `vec1` includes genera names which miss specific epithet, this object lists all the species in `vec2` belonging to each of the genera.

$subspecies if `vec1` includes subspecies (i.e. two epithets after genus name), this object lists species in `vec2` possibly corresponding to each of the subspecies.

$epithet lists species with matching epithets as possible synonyms.

$misspelling lists possible misspelled names. For each proposed mismatched names pair the proportion of characters in the `vec1` differing from the string in `vec2` is returned.

Author(s)

Silvia Castiglione, Carmela Serio, Antonella Esposito

Examples

```
## Not run:
names(DataFelids$statefel)->nams
nams[c(19,12,37,80,43)]<-c("Puma_yagouaroundi","Felis_manul","Catopuma",
                           "Pseudaelurus","Panthera_zdansky")
nams<-nams[-81]

namesCompare(nams,names(DataFelids$statefel))
namesCompare(names(DataFelids$statefel),nams)

## End(Not run)
```

---

node.paths   *Tracing nodes along paths*

Description

Given a vector of nodes, the function collates nodes along individual lineages from the youngest (i.e. furthest from the tree root) to the oldest.

Usage

```
node.paths(tree, vec)
```

Arguments

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

vec   a vector of node numbers

Value

A list of node paths, each starting from the youngest node (i.e. furthest from the tree root) and ending to the oldest along the path.

Author(s)

Silvia Castiglione, Pasquale Raia

Examples

```
require(ape)

rtree(100)->tree
sample(seq(Ntip(tree)+1,Ntip(tree)+Nnode(tree)),20)->nods
plot(tree,show.tip.label=FALSE)
nodelabels(node=nods,frame="n",col="red")
node.paths(tree=tree, vec=nods)
```

---

overfitRR   *Testing RRphylo methods overfit*

Description

Testing the robustness of `search.trend` (*Castiglione et al. 2019a*), `search.shift` (*Castiglione et al. 2018*), `search.conv` (*Castiglione et al. 2019b*), and `PGLS_fossil` results to sampling effects and phylogenetic uncertainty.

Usage

```
overfitRR(RR,y,phylo.list=NULL,s=0.25,swap.args=NULL,trend.args=NULL,shift.args=NULL,
conv.args=NULL, pgls.args=NULL,aces=NULL,x1=NULL,aces.x1=NULL,cov=NULL,
rootV=NULL,nsim=100,clus=.5)
```

Arguments

RR	an object produced by [RRphylo]().
y	a named vector of phenotypes.
phylo.list	a list (or multiPhylo) of alternative phylogenies to be tested.
s	the percentage of tips to be cut off. It is set at 25% by default. If phylo.list is provided, this argument is ignored.
swap.args	a list of arguments to be passed to the function [swapONE](), including list(si=NULL,si2=NULL, node=NULL). If swap.arg is unspecified, the function automatically sets both si and si2 to 0.1. If phylo.list is provided, swapping is not performed.
trend.args	a list of arguments specific to the function search.trend, including list(node=NULL,x1.residuals=FA If a trend for the whole tree is to be tested, type trend.args = list(). No trend is tested if left unspecified.
shift.args	a list of arguments specific to the function search.shift, including list(node=NULL, state=NULL). Arguments node and state can be specified at the same time.
conv.args	a list of arguments specific to the function search.conv, including list(node=NULL, state=NULL,declust=FALSE). Arguments node and state can be specified at the same time.
pgls.args	a list of arguments specific to the function PGLS_fossil, including list(modform, data,tree=FALSE,RR=TRUE,...). If tree=TRUE, PGLS_fossil is performed by using the RRphylo output tree as tree argument. If RR=TRUE, PGLS_fossil is performed by using the RRphylo output as RR argument. Arguments tree and RR can be TRUE at the same time. ... are further argument passed to PGLS_fossil.
aces	if used to produce the RR object, the vector of those ancestral character values at nodes known in advance must be specified. Names correspond to the nodes in the tree.
x1	the additional predictor to be specified if the RR object has been created using an additional predictor (i.e. multiple version of RRphylo). 'x1' vector 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 predictor as well, and taking the vector of ancestral states and tip values to form the x1.
aces.x1	a named vector of ancestral character values at nodes for x1. It must be indicated if the RR object has been created using both aces and x1. Names correspond to the nodes in the tree.
cov	if used to produce the RR object, the covariate must be specified. As in RRphylo, the covariate vector 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.

rootV	if used to produce the RR object, the phenotypic value at the tree root must be specified.
nsim	number of simulations to be performed. It is set at 100 by default.
clus	the proportion of clusters to be used in parallel computing. To run the single-threaded version of overfitRR set clus = 0.

Details

Methods using a large number of parameters risk being overfit. This usually translates in poor fitting with data and trees other than the those originally used. With RRphylo methods this risk is usually very low. However, the user can assess how robust the results got by applying search.shift, search.trend, search.conv or PGLS_fossil are by running overfitRR. With the latter, the original tree and data are subsampled by specifying a s parameter, that is the proportion of tips to be removed from the tree. In some cases, though, removing as many tips as imposed by s would delete too many tips right in clades and/or states under testing. In these cases, the function maintains no less than 5 species at least in each clade/state under testing (or all species if there is less), reducing the sampling parameter s if necessary. Internally, overfitRR further shuffles the tree by using the function [swapONE](). Thereby, both the potential for overfit and phylogenetic uncertainty are accounted for straight away.

Otherwise, a list of alternative phylogenies can be supplied to overfitRR. In this case subsampling and swapping arguments are ignored, and robustness testing is performed on the alternative topologies as they are. If a clade has to be tested either in search.shift, search.trend, or search.conv, the function scans each alternative topology searching for the corresponding clade. If the species within such clade on the alternative topology differ more than 10 species within the clade in the original tree, the identity of the clade is considered disrupted and the test is not performed.

Value

The function returns a 'RRphyloList' object containing:

$mean.sampling the mean proportion of species actually removed from the tree over the iterations.

$tree.list a 'multiPhylo' list including the trees generated within overfitRR

$RR.list a 'RRphyloList' including the results of each RRphylo performed within overfitRR

$rootCI the 95% confidence interval around the root value.

$ace.regressions a 'RRphyloList' including the results of linear regression between ancestral state estimates before and after the subsampling.

$conv.results a list including results for search.conv performed under clade and state conditions. If a node pair is specified within conv.args, the $clade object contains the percentage of simulations producing significant p-values for convergence between the clades, and the proportion of tested trees (i.e. where the clades identity was preserved; always 1 if no phylo.list is supplied). If a state vector is supplied within conv.args, the object $state contains the percentage of simulations producing significant p-values for convergence within (single state) or between states (multiple states).

$shift.results a list including results for search.shift performed under clade and sparse conditions. If one or more nodes are specified within shift.args, the $clade object contains for each node the percentage of simulations producing significant p-value separated by shift sign, and the

same figures by considering all the specified nodes as evolving under a single rate (all.clades). For each node the proportion of tested trees (i.e. where the clade identity was preserved; always 1 if no `phylo.list` is supplied) is also indicated. If a state vector is supplied within `shift.args`, the object `$sparse` contains the percentage of simulations producing significant p-value separated by shift sign ($p.states).

$trend.results a list including the percentage of simulations showing significant p-values for phenotypes versus age and absolute rates versus age regressions for the entire tree separated by slope sign ($tree). If one or more nodes are specified within `trend.args`, the list also includes the same results at nodes ($node) and the results for comparison between nodes ($comparison). For each node the proportion of tested trees (i.e. where the clade identity was preserved; always 1 if no `phylo.list` is supplied) is also indicated.

$pgls.results two 'RRphyloList' objects including results of PGLS_fossil performed by using the phylogeny as it is ($tree) or rescaled according to the RRphylo rates ($RR).

Author(s)

Silvia Castiglione, Carmela Serio, Pasquale Raia

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*, 9: 974-983.doi:10.1111/2041-210X.12954

Castiglione, S., Serio, C., Mondanaro, A., Di Febbraro, M., Profico, A., Girardi, G., & Raia, P. (2019a) Simultaneous detection of macroevolutionary patterns in phenotypic means and rate of change with and within phylogenetic trees including extinct species. *PLoS ONE*, 14: e0210101. https://doi.org/10.1371/journal.pone.0210101

Castiglione, S., Serio, C., Tamagnini, D., Melchionna, M., Mondanaro, A., Di Febbraro, M., Profico, A., Piras, P.,Barattolo, F., & Raia, P. (2019b). A new, fast method to search for morphological convergence with shape data. *PLoS ONE*, 14, e0226949. https://doi.org/10.1371/journal.pone.0226949

See Also

overfitRR vignette ; search.trend vignette ; search.shift vignette ; search.conv vignette ;

Examples

```
## Not run:
data("DataOrnithodirans")
DataOrnithodirans$treedino->treedino
DataOrnithodirans$massdino->massdino
DataOrnithodirans$statedino->statedino
cc<- 2/parallel::detectCores()

# Extract Pterosaurs tree and data
library(ape)
extract.clade(treedino,746)->treeptero
massdino[match(treeptero$tip.label,names(massdino))]->massptero
massptero[match(treeptero$tip.label,names(massptero))]->massptero
```

```
RRphylo(tree=treedino,y=massdino)->dinoRates
RRphylo(tree=treeptero,y=log(massptero))->RRptero

# Case 1 search.shift under both "clade" and "sparse" condition
search.shift(RR=dinoRates, status.type= "clade")->SSnode
search.shift(RR=dinoRates, status.type= "sparse", state=statedino)->SSstate

overfitRR(RR=dinoRates,y=massdino,swap.args =list(si=0.2,si2=0.2),
          shift.args = list(node=rownames(SSnode$single.clades),state=statedino),
          nsim=10,clus=cc)->orr.ss

# Case 2 search.trend on the entire tree
search.trend(RR=RRptero, y=log(massptero),nsim=100,clus=cc,cov=NULL,node=NULL)->STtree

overfitRR(RR=RRptero,y=log(massptero),swap.args =list(si=0.2,si2=0.2),
          trend.args = list(),nsim=10,clus=cc)->orr.st1

# Case 3 search.trend at specified nodescov=NULL,
search.trend(RR=RRptero, y=log(massptero),node=143,clus=cc)->STnode

overfitRR(RR=RRptero,y=log(massptero),
          trend.args = list(node=143),nsim=10,clus=cc)->orr.st2

# Case 4 overfitRR on multiple RRphylo
data("DataCetaceans")
DataCetaceans$treecet->treecet
DataCetaceans$masscet->masscet
DataCetaceans$brainmasscet->brainmasscet
DataCetaceans$aceMyst->aceMyst

ape::drop.tip(treecet,treecet$tip.label[-match(names(brainmasscet),
                                                treecet$tip.label)])->treecet.multi
masscet[match(treecet.multi$tip.label,names(masscet))]->masscet.multi

RRphylo(tree=treecet.multi,y=masscet.multi)->RRmass.multi
RRmass.multi$aces[,1]->acemass.multi
c(acemass.multi,masscet.multi)->x1.mass

RRphylo(tree=treecet.multi,y=brainmasscet,x1=x1.mass)->RRmulti
search.trend(RR=RRmulti, y=brainmasscet,x1=x1.mass,clus=cc)->STcet
overfitRR(RR=RRmulti,y=brainmasscet,trend.args = list(),
          x1=x1.mass,nsim=10,clus=cc)->orr.st3

search.trend(RR=RRmulti, y=brainmasscet,x1=x1.mass,x1.residuals=TRUE,
             clus=cc)->STcet.resi
overfitRR(RR=RRmulti,y=brainmasscet,trend.args = list(x1.residuals=TRUE),
          x1=x1.mass,nsim=10,clus=cc)->orr.st4

# Case 5 searching convergence between clades and within a single state
data("DataFelids")
DataFelids$PCscoresfel->PCscoresfel
```

```
DataFelids$treefel->treefel
DataFelids$statefel->statefel

RRphylo(tree=treefel,y=PCscoresfel,clus=cc)->RRfel
search.conv(RR=RRfel, y=PCscoresfel, min.dim=5, min.dist="node9",clus=cc)->SC.clade
as.numeric(c(rownames(SC.clade[[1]])[1],as.numeric(as.character(SC.clade[[1]][1,1]))))->conv.nodes

overfitRR(RR=RRfel, y=PCscoresfel,conv.args =
list(node=conv.nodes,state=statefel,declust=TRUE),nsim=10,clus=cc)->orr.sc

# Case 6 overfitRR on PGLS_fossil
library(phytools)
rtree(100)->tree
fastBM(tree)->resp
fastBM(tree,nsim=3)->resp.multi
fastBM(tree)->pred1
fastBM(tree)->pred2

PGLS_fossil(modform=y1~x1+x2,data=list(y1=resp,x2=pred1,x1=pred2),tree=tree)->pgls_noRR

RRphylo::RRphylo(tree,resp)->RR
PGLS_fossil(modform=y1~x1+x2,data=list(y1=resp,x2=pred1,x1=pred2),tree=tree,RR=RR)->pgls_RR

overfitRR(RR=RR,y=resp,
          pgls.args=list(modform=y1~x1+x2,data=list(y1=resp,x2=pred1,x1=pred2),
                         tree=TRUE,RR=TRUE),nsim=10,clus=cc)->orr.pgls1

PGLS_fossil(modform=y1~x1+x2,data=list(y1=resp.multi,x2=pred1,x1=pred2),tree=tree)->pgls2_noRR
cc<- 2/parallel::detectCores()
RRphylo::RRphylo(tree,resp.multi,clus=cc)->RR
PGLS_fossil(modform=y1~x1+x2,data=list(y1=resp.multi,x2=pred1,x1=pred2),tree=tree,RR=RR)->pgls2_RR

overfitRR(RR=RR,y=resp.multi,
          pgls.args=list(modform=y1~x1+x2,data=list(y1=resp.multi,x2=pred1,x1=pred2),
                         tree=TRUE,RR=TRUE),nsim=10,clus=cc)->orr.pgls2


## End(Not run)
```

PGLS_fossil	*Phylogenetic Generalized Least Square with phylogenies including fossils*

Description

The function performs pgls for non-ultrametric trees using a variety of evolutionary models or [RRphylo](#) rates to change the tree correlation structure.

Usage

```
PGLS_fossil(modform,data,tree=NULL,RR=NULL,...)
```

Arguments

modform	the formula for the regression model.
data	a list of named vectors including response and predictor variables as named in modform.
tree	a phylogenetic tree to be indicated for any model other than RRphylo is used to rescale tree branches. The tree needs not to be ultrametric and fully dichotomous.
RR	the result of RRphylo performed on the response variable. If RR is specified, tree branches are rescaled to the absolute branch-wise rate values calculated by RRphylo to transform the variance-covariance matrix.
...	further argument passed to the function phylolm.

Value

Fitted pgls parameters and significance.

Author(s)

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

See Also

RRphylo vignette

Examples

```
## Not run:
library(ape)
library(phytools)

rtree(100)->tree
fastBM(tree)->resp
fastBM(tree,nsim=3)->resp.multi
fastBM(tree)->pred1
fastBM(tree)->pred2

PGLS_fossil(modform=y1~x1+x2,data=list(y1=resp,x2=pred1,x1=pred2),tree=tree)->pgls_noRR

RRphylo::RRphylo(tree,resp)->RR
PGLS_fossil(modform=y1~x1+x2,data=list(y1=resp,x2=pred1,x1=pred2),tree=tree,RR=RR)->pgls_RR

# To derive log-likelihood and AIC for PGLS_fossil outputs the function AIC can be applied
AIC(pgls_noRR)
AIC(pgls_RR)
```

```
PGLS_fossil(modform=y1~x1+x2,data=list(y1=resp.multi,x2=pred1,x1=pred2),tree=tree)->pgls2_noRR
cc<- 2/parallel::detectCores()
RRphylo::RRphylo(tree,resp.multi,clus=cc)->RR
PGLS_fossil(modform=y1~x1+x2,data=list(y1=resp.multi,x2=pred1,x1=pred2),tree=tree,RR=RR)->pgls2_RR

AIC(pgls2_noRR)
AIC(pgls2_RR)

## End(Not run)
```

---

phyloclust	*Test for phylogenetic clustering*

Description

The function tests the presence of phylogenetic clustering for species within a focal state.

Usage

```
phyloclust(tree,state,focal,nsim=100)
```

Arguments

tree	a phylogenetic tree. The tree needs not to be ultrametric or fully dichotomous.
state	the named vector of tip states.
focal	the focal state to be tested for phylogenetic clustering.
nsim	number of simulations to perform the phylogenetic clustering test.

Details

To test for phylogenetic clustering, the function computes the mean cophenetic (i.e. evolutionary time) distance between all the species under the focal state. Such value is compared to a random distribution of time distances obtained by sampling nsim times as many random tips as those under the focal state. In the presence of significant phylogenetic clustering, tips under the focal state are randomly removed until the p becomes >0.05 or only 3 tips are left.

Value

The function returns a list including the p-value ($p) for the test of phylogenetic clustering and a $declusterized object containing the declusterized versions of the original tree and state vector (i.e. tips are removed as to make p>0.05) and the vector of removed species.

Author(s)

Silvia Castiglione, Pasquale Raia

Examples

```
data("DataFelids")
DataFelids$treefel->treefel
DataFelids$statefel->statefel

phyloclust(tree=treefel,state=statefel,focal="saber")
```

---

plotConv   *Graphical representation of search.conv results*

Description

This function generates customized functions to produce plots out of [search.conv](#) results.

Usage

```
plotConv(SC,y,variable,RR=NULL,state=NULL,aceV=NULL)
```

Arguments

SC	an object produced by [search.conv](#).
y	the multivariate phenotype used to perform search.conv.
variable	the index of result to plot. If convergence between clades is inspected, this is the position within the SC$'average distance from group centroids' vector of the clade pair to be plotted. In the case of convergence between states, this is the number of the line of SC$state.res where results for the state pair are returned.
RR	the object produced by [RRphylo](#) used to perform search.conv. This is not indicated if convergence between states is tested.
state	the named vector of tip states used to perform search.conv. This is not indicated if convergence between clades is tested.
aceV	phenotypic values at internal nodes to be provided if used to perform search.conv.

Value

If convergence between clades was tested, plotConv returns a list of four functions:

$plotHistTips shows the mean Euclidean distance computed between phenotypic vectors of all the tips belonging to the converging clades as compared to the distribution of distances between all possible pair of tips across the rest of the tree. The usage is: ...$plotHistTips(hist.args=NULL,line.args=NULL), where hist.args is a list of further arguments passed to the function hist, and line.args is a list of further arguments passed to the function lines.

$plotHistAces shows the Euclidean distance computed between phenotypic vectors of the MRCAs of the converging clades as compared to the distribution of distances between all possible pairs of nodes across the rest of the tree. The usage is identical to $plotHistTips.

`$plotPChull` generates a PC1/PC2 plot obtained by performing a PCA of the species phenotypes. Convergent clades are indicated by colored convex hulls. Large dots represent the mean phenotypes per clade (i.e. their group centroids) and asterisks (customizable) represent the ancestral phenotypes of the individual clades. The usage is: `...$plotPChull(plot.args=NULL,chull.args=NULL,means.args=NULL,ace.args=NULL,legend.args=list()`,where `plot.args` is a list of further arguments passed to the function `plot`, `chull.args` is a list of further arguments passed to the function `polygon`, `means.args` and `ace.args` are lists of further argument passed to the function `points` to customize the dots representing the centroids and the ancestral phenotypes respectively, and `legend.args` is a list of additional arguments passed to the function `legend` (if = NULL the legend is not plotted).

`$plotTraitgram` produces a modified traitgram plot (see package **picante**) highlighting the branches of the clades found to converge. The usage is: `plotTraitgram(colTree=NULL,colNodes=NULL,...)`, where `colTree` is the color to represent the traitgram lines not pertaining the converging clades, `colNodes` is the color (or the vector of colors) to represent the traitgram lines pertaining the converging clades, and `...` are further arguments passed to the function `plot` to plot lines.

If convergence between states was tested, `plotConv` returns a list of two functions:

`$plotPChull` generates a PC1/PC2 plot obtained by performing a PCA of the species phenotypes, with colored convex hulls enclosing species belonging to different states. The usage is: `...$plotPChull(plot.args=NULL,chull.args=NULL,points.args=NULL, legend.args=list()`, where `plot.args` is a list of further arguments passed to the function `plot`, `chull.args` is a list of further arguments passed to the function `polygon`, `points.args` is a list of further argument passed to the function `points`, and `legend.args` is a list of additional arguments passed to the function `legend` (if = NULL the legend is not plotted).

`$plotPolar` produces a polar plot of the mean angle within/between state/s as compared to the 95 angles. The usage is: `...$plotPolar(polar.args=NULL,polygon.args=NULL,line.args=NULL)`, where `polar.args` is a list of further arguments passed to the function [polar.plot](#) to set the plot basics (i.e. `radial.lim`, `start`, and so on), `polygon.args` is a list of further arguments passed to the function `polar.plot` under the condition `rp.type="p"` (see `plotrix::polar.plot` for details) to set the angles distribution graphics, and `line.args` is a list of further arguments passed to the function `polar.plot` under the condition `rp.type="r"` to set the mean angle graphics.

Author(s)

Silvia Castiglione

References

Castiglione, S., Serio, C., Tamagnini, D., Melchionna, M., Mondanaro, A., Di Febbraro, M., Profico, A., Piras, P.,Barattolo, F., & Raia, P. (2019). A new, fast method to search for morphological convergence with shape data. *PLoS ONE*, 14, e0226949. https://doi.org/10.1371/journal.pone.0226949

See Also

[search.conv vignette](#)

[plotConv vignette](#)

Examples

```
## Not run:
data("DataFelids")
DataFelids$PCscoresfel->PCscoresfel
DataFelids$treefel->treefel
DataFelids$statefel->statefel->state2
state2[sample(which(statefel=="nostate"),20)]<-"st2"
cc<- 2/parallel::detectCores()

RRphylo(treefel,PCscoresfel,clus=cc)->RRfel

search.conv(RR=RRfel, y=PCscoresfel, min.dim=5, min.dist="node9",clus=cc)->sc_clade
plotConv(sc_clade,PCscoresfel,variable=2,RR=RRfel)->pc

pc$plotHistTips(hist.args = list(col="gray80",yaxt="n",cex.axis=0.8,cex.main=1.5),
                line.args = list(lwd=3,lty=4,col="purple"))
pc$plotHistAces(hist.args = list(col="gray80",cex.axis=0.8,cex.main=1.5),
                line.args = list(lwd=3,lty=4,col="gold"))
pc$plotPChull(chull.args = list(border=c("cyan","magenta"),lty=1),
              means.args = list(pch=c(23,22),cex=3,bg=c("cyan2","magenta2")),
              ace.args=list(pch=9),legend.args = NULL)
pc$plotTraitgram(colTree = "gray70",colNodes = c("cyan","magenta"))


search.conv(tree=treefel, y=PCscoresfel, state=statefel,declust=TRUE,clus=cc)->sc_state
plotConv(sc_state,PCscoresfel,variable=1,state=statefel)->pc
pc$plotPChull(chull.args = list(border=c("gray70","blue"),lty=1),
              points.args = list(pch=c(23,22),bg="gray"),
              legend.args = list(pch=c(23,22),x="top"))

pc$plotPolar(polar.args = list(clockwise=TRUE,start=0,rad.col="black",grid.col="black"),
             polygon.args = list(line.col="green",poly.col=NA,lwd=2),
             line.args = list(line.col="deeppink",lty=2,lwd=3))


## End(Not run)
```

---

plotRates	*Plot RRphylo rates at a specified node*

Description

This function generates customized functions to produce histograms and lollipop charts of 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=NULL,filename=NULL)
```

Arguments

RR	an object produced by RRphylo.
node	the node subtending the clade of interest.
export.tiff	is deprecated.
filename	is deprecated.

Value

The function returns a list of functions:

$plotHist returns the histograms of rates (in ln absolute values) computed for the focal clade against rates computed the rest of the tree. The usage is: ...$plotHist(hist.args=list(col1,col2),legend.args=list()), where legend.args is a list of additional arguments passed to the function legend (if = NULL the legend is not plotted) and hist.args is a list of further arguments passed to the function plot.histogram. hist.args default arguments include histogram colors for background rates (col1) and rates of the clade under inspection (col2).

$plotLollipop returns the lollipop chart of the rates of individual branches of the focal clade collated in increasing rate value, and contrasted to the average rate computed over the rest of the tree branches (the vertical line). The usage is: ...$plotLollipop(lollipop.args=NULL,line.args=NULL), where lollipop.args is a list of further arguments passed to the function lollipoPlot and line.args is a list of additional arguments passed to the function line. This function additionally returns the vector of rates for the focal clade, collated in increasing order.

Author(s)

Silvia Castiglione, Pasquale Raia

See Also

RRphylo vignette

RRphylo vignette

plotRates vignette

Examples

```
data("DataApes")
DataApes$PCstage->PCstage
DataApes$Tstage->Tstage
cc<- 2/parallel::detectCores()

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

plotRates(RR,node=72)->pR
pR$plotHist(hist.args=list(col1="cyan1",col2="blue"),legend.args=list(x="topright"))
pR$plotLollipop(lollipop.args=list(col="chartreuse",bg="chartreuse"),
                line.args=list(col="deeppink",lwd=2))
```

---

plotShift   *Graphical representation of search.shift results*

Description

This function generates customized functions to produce plots out of search.shift results.

Usage

```
plotShift(RR,SS,state=NULL)
```

Arguments

RR   the object produced by RRphylo used to perform search.shift.

SS   an object produced by search.shift.

state   if search.shift was performed under status.type = "sparse", this is the
            same state vector provided to the function.

Value

The function returns a function to generate a customizable plot of search.shift results.

If search.shift was performed under status.type = "clade", plotShift returns a $plotClades
function which highlights the shifting clades onto the phylogenetic tree. The usage is: ...$plotClades(tree.args=NULL,sy
where tree.args is a list of further arguments passed to the function plot.phylo, and symbols.args
is a list of further arguments passed to the function symbols (n.b. the shape of the symbol is
not customizable). The function automatically plots red circles for negative shifts and blue cir-
cles for positive shifts, in each cases with the radium proportional to the absolute value of rate
difference. The user can choose different color options for positive/negative shifts by setting
symbols.args=list(fg=c(pos="color for positive shift",neg="color for negative shift")),
or provide a vector of as many colors as the number of shifting clades. The same applies to the ar-
gument "bg".

If search.shift was performed under status.type = "sparse", plotShift returns a $plotStates
function which plots the states onto the phylogenetic trees. The usage is: ...$plotStates(tree.args=NULL,points.args=
where tree.args is a list of further arguments passed to the function plot.phylo, points.args
is a list of further arguments passed to the function points, and legend.args is a list of additional
arguments passed to the function legend (if = NULL the legend is not plotted). If as many colors/pch
values as the number of different states are provided in points.args, each of them is assigned to
each states taken in the same order they are returned by search.shift.

Author(s)

Silvia Castiglione

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*, 9: 974-983.doi:10.1111/2041-210X.12954

See Also

Examples

```
## Not run:
data("DataOrnithodirans")
DataOrnithodirans$treedino->treedino
DataOrnithodirans$massdino->massdino
DataOrnithodirans$statedino->statedino

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

search.shift(RR=dinoRates,status.type="clade")->SSauto
plotShift(RR=dinoRates,SS=SSauto)->plotSS
plotSS$plotClades()

search.shift(RR=dinoRates,status.type="clade",node=c(696,746))->SSnode
plotShift(RR=dinoRates,SS=SSnode)->plotSS
plotSS$plotClades(tree.args=list(no.margin=TRUE),
                  symbols.args=list(fg=NA,bg=c(pos="cyan",neg="magenta")))


search.shift(RR=dinoRates,status.type= "sparse",state=statedino)->SSstate
plotShift(RR=dinoRates,SS=SSstate,state=statedino)->plotSS
plotSS$plotStates(tree.args=list(no.margin=TRUE),
                  points.args=list(bg=c("gold","forestgreen","royalblue","white"),
                                   col=c("black","black","black","orangered"),
                                   pch=c(21,22,24,11)),legend.args=list())

## End(Not run)
```

| plotTrend | *Graphical representation of search.trend results* |

Description

This function generates customized functions to produce plots of phenotype versus time and absolute evolutionary rates versus time regressions for the entire phylogeny and individual clades. Each custom function takes as first argument the index or name of the variable (as in the search.trend output in $trends.data$phenotypeVStime) to be plotted.

Usage

```
plotTrend(ST)
```

Arguments

ST   an object produced by search.trend.

Value

The function returns a list of functions:

$plotSTphen returns the plot of rescaled phenotype versus age regression. The 95% confidence intervals of slopes produced by regressing phenotypes simulated under the Brownian motion are plotted as a polygon. The usage is . . . $plotSTphen(variable,plot.args=NULL,polygon.args=NULL,line.args=NULL), where variable is the index or name of the variable to be plotted, plot.args is a list of further arguments passed to the function plot, polygon.args is a list of further arguments passed to the function polygon, and line.args is a list of further arguments passed to the function lines. The functions automatically plots white points for internal nodes, red points for tips, a blue regression line, and a gray shaded polygon to represent the 95% confidence intervals of Brownian motion slopes.

$plotSTrates returns the plot of log rescaled rates versus age regression. The 95% confidence intervals of slopes produced by regressing rates simulated under the Brownian motion are plotted as a shaded area. The arguments are the same as described for $plotSTphen. In the case of multivariate y, the 2-norm vector of multiple rates (see [RRphylo](#) for details) can be plotted by setting variable = "rate" or variable = the number of actual variables + 1.

$plotSTphenNode returns plots of rescaled phenotype versus age regression for individual clades. The usage is . . . $plotSTphenNode(variable,node,plot.args=NULL,lineTree.args=NULL, lineNode.args=NULL,noc where variable is the same as plotSTphen and node is the vector of indices or numbers (as inputted to search.trend, to be indicated as character) of nodes to be plotted. The function allows up to nine nodes at the same time. plot.args is a list of further arguments passed to the function plot, including pch.node, a custom argument to set individual pch at nodes (its length can be one or as long as the number of nodes). lineTree.args is a list of further arguments passed to the function lines used to plot the regression line for the entire tree lineNode.args is a list of further arguments passed to the function lines used to plot the regression line for individual nodes. node.palette is the vector of colors specific to nodes points and lines. Its length can be one or as long as the number of nodes.

$plotSTratesNode returns plots of absolute rates versus age regression for individual clades. The arguments are the same as described for $plotSTphenNode. In the case of multivariate y, the 2-norm vector of multiple rates (see [RRphylo](#) for details) can be plotted by setting variable = "rate" or variable = the number of actual variables + 1.

Author(s)

Silvia Castiglione, Carmela Serio

References

Castiglione, S., Serio, C., Mondanaro, A., Di Febbraro, M., Profico, A., Girardi, G., & Raia, P. (2019) Simultaneous detection of macroevolutionary patterns in phenotypic means and rate of change with and within phylogenetic trees including extinct species. *PLoS ONE*, 14: e0210101. https://doi.org/10.1371/journal.pone.0210101

See Also

[search.trend](#) vignette

[plotTrend](#) vignette

Examples

```
 ## Not run:
data("DataOrnithodirans")
DataOrnithodirans$treedino->treedino
DataOrnithodirans$massdino->massdino
cc<- 2/parallel::detectCores()

# Extract Pterosaurs tree and data
library(ape)
extract.clade(treedino,746)->treeptero
massdino[match(treeptero$tip.label,names(massdino))]->massptero
massptero[match(treeptero$tip.label,names(massptero))]->massptero

RRphylo(tree=treeptero,y=log(massptero))->RRptero

search.trend(RR=RRptero, y=log(massptero), nsim=100, node=143, clus=cc,cov=NULL)->ST

plotTrend(ST)->plotST

plotST$plotSTphen(1) # to plot phenotypic trend through time for entire tree
plotST$plotSTphen("log(massptero)",plot.args=list(cex=1.2,col="blue")) # equivalent to above

plotST$plotSTrates(1) # to plot rates trend through time for entire tree

# to plot phenotypic trend through time for the clade
plotST$plotSTphenNode("log(massptero)",plot.args=list(xlab="Age",main="Phenotypic trend"),
                     lineTree.args=list(lty=1,col="pink"),lineNode.args=list(lwd=3),
                     node.palette=c("chartreuse"))

# to plot rates trend through time for the clade
plotST$plotSTratesNode("rate")


## End(Not run)
```

---

random.evolvability.test

Randomization test for phylogenetic structuring in evolvability

---

Description

The function is a wrapper around the function MeanMatrixStatistics from the package **evolqg** (*Melo et al. 2015*). It estimates ancestral character at internal nodes either according to Brownian Motion or by means of RRphylo (see the argument node.estimation), then performs MeanMatrixStatistics to calculate: Mean Squared Correlation, ICV, Autonomy, ConditionalEvolvability, Constraints, Evolvability, Flexibility, Pc1Percent, and Respondability. To assess the importance of phylogenetic structuring (signal) on Respondability Evolvability, and Flexibility. The function performs a randomization test by randomly shuffling the species on tree and replicating the analyses nsim times. A p-value is computed by contrasting the real metrics to the ones derived by randomization.

Usage

```
random.evolvability.test(tree,data,node.estimation=c("RR","BM"),
  aces=NULL,iterations=1000,nsim=100,clus=0.5)
```

Arguments

tree	a phylogenetic tree. The tree needs not to be ultrametric or fully dichotomous.
data	a matrix or data.frame of phenotypic data having species as rownames
node.estimation	
	specify the method to compute ancestral character at nodes. It can be one of "RR", to compute ancestral states by mean of RRphylo, or "BM", to use **phytools**' function fastAnc (*Paradis & Schliep 2019*) to estimate ancestral characters at nodes according to Brownian Motion.
aces	a named matrix of known ancestral character values at nodes. Names correspond to the nodes in the tree.
iterations	the iterations argument to be indicated in MeanMatrixStatistics
nsim	the number of simulations to be performed for the randomization test, by default nrep is set at 100.
clus	the proportion of clusters to be used in parallel computing. To run the single-threaded version of NOME set clus = 0.

Value

The function returns a list object including ($means) the mean values for all statistics as produced by MeanMatrixStatistics and ($means) the significance levels for Respondability, Evolvability, and Flexibility.

Author(s)

Silvia Castiglione, Gabriele Sansalone, Pasquale Raia

References

Melo, D., Garcia, G., Hubbe, A., Assis, A. P., & Marroig, G. (2015). EvolQG-An R package for evolutionary quantitative genetics. *F1000Research*, 4.

Revell, L. J. (2012) phytools: An R package for phylogenetic comparative biology (and other things). *Methods in Ecology and Evolution*, 3, 217-223.

See Also

[RRphylo vignette](#)

Examples

```
## Not run:
library(ape)
library(phytools)

rtree(30)->tree
fastBM(tree,nsim=4)->y

random.evolvability.test(tree=tree,data=y,node.estimation="RR")->rEvTest


## End(Not run)
```

---

rate.map   *Mapping rate and direction of phenotypic change on 3D surfaces.*

Description

Given vectors of RW (or PC) scores, the function selects the RW(PC) axes linked to highest (and lowest) evolutionary rate values and reconstructs the morphology weighted on them. In this way, rate.map shows where and how the phenotype changed the most between any pair of taxa.

Usage

```
rate.map(x, RR, PCscores, pcs, mshape, out.rem = TRUE,
  shape.diff=FALSE, show.names=TRUE)
```

Arguments

x	the species/nodes to be compared; it can be a single species, or a vector containing two species or a species and any of its parental nodes.
RR	an object generated by using the [RRphylo](#) function
PCscores	PC scores.
pcs	RW (or PC) vectors (eigenvectors of the covariance matrix) of all the samples.
mshape	the Consensus configuration.
out.rem	logical: if TRUE mesh triangles with outlying area difference are removed.
shape.diff	logical: if TRUE, the mesh area differences are displayed in an additional 3d plot.
show.names	logical: if TRUE, the names of the species are displayed in the 3d plot.

Details

After selecting the PC axes, rate.map automatically builds a 3D mesh on the mean shape calculated from the Relative Warp Analysis (RWA) or Principal Component Analysis (PCA) (*Schlager 2017*) by applying the function [vcgBallPivoting](#) (**Rvcg**). Then, it compares the area differences between corresponding triangles of the 3D surfaces reconstructed for the species and surface of the mrca. Finally, rate.map returns a 3D plot showing such comparisons displayed on shape of the mrca used

as the reference.The colour gradient goes from blue to red, where blue areas represent expansion of the mesh, while the red areas represent contractions of the mesh triangles. In the calculation of the differences of areas we supply the possibility to find and remove outliers from the vectors of areas calculated on the reference and target surfaces. We suggest considering this possibility if the mesh may contain degenerate facets. Additionally, `rate.map` allows to investigate the pure morphological comparison of shapes by excluding the evolutionary rate component by setting the argument `show.diff = TRUE`. In this case, a second 3D plot will be displayed highlighting area differences in terms of expansion (green) and contraction (yellow).

Value

The function returns a list including:

- **$selected** a list of PCs axes selected for higher evolutionary rates for each species.
- **$surfaces** a list of reconstructed coloured surfaces of the given species and of the most recent common ancestor.

Author(s)

Marina Melchionna, Antonio Profico, Silvia Castiglione, Gabriele Sansalone, Pasquale Raia

References

Schlager, S. (2017). *Morpho and Rvcg-Shape Analysis in R: R-Packages for geometric morphometrics, shape analysis and surface manipulations.* In: Statistical shape and deformation analysis. Academic Press. Castiglione, S., Melchionna, M., Profico, A., Sansalone, G., Modafferi, M., Mondanaro, A., Wroe, S., Piras, P., & Raia, P. (2021). Human face-off: a new method for mapping evolutionary rates on three-dimensional digital models. Palaeontology. doi:10.1111/pala.12582

See Also

RRphylo vignette ; relWarps ; procSym

Examples

```
## Not run:
data(DataSimians)
DataSimians$pca->pca
DataSimians$tree->tree
dato<-pca$PCscores

RRphylo(tree,dato)->RR

Rmap<-rate.map(x=c("Pan_troglodytes","Gorilla_gorilla"),RR=RR, PCscores=dato,
               pcs=pca$PCs, mshape=pca$mshape, shape.diff = TRUE)


## End(Not run)
```

---

rescaleRR   *Rescaling phylogenetic trees*

Description

The function rescales all branches and leaves of the phylogenetic tree.

Usage

```
rescaleRR(tree,RR=NULL,height=NULL,trend=NULL,delta=NULL,kappa=NULL,lambda=NULL)
```

Arguments

tree	the phylogenetic tree to be rescaled.
RR	is the output of RRphylo performed on tree. If this parameter is indicated, the tree branches are rescaled according to branch-wise phenotypic evolutionary rates fitted by [RRphylo](). When a multivariate phenotype is used, rescaling is operated on the norm-2 vector of rates.
height	is the desired height of the tree. If this parameter is indicated, the tree branches are rescaled to match the total height.
trend	is a diffusion model with linear trend in rates through time. The trend scaling is largely based on package **geiger**'s rescale.phylo function.
delta	if this parameter is indicated, the tree is rescaled according to Pagel's delta transform (Pagel 1999). Nodes are pushed toward the present for values of delta ranging between 0 and 1. The converse applies for delta larger than 1. Negative delta values are not allowed.
kappa	if this parameter is indicated, the tree is rescaled according to Pagel's kappa transform (Pagel 1999). At kappa = 1 the tree is left unmodified. Branches become increasingly closer to 1 as kappa approaches 0, making evolution independent from branch lengths. Negative kappa values are not allowed.
lambda	if this parameter is indicated, the tree is rescaled according to Pagel's lambda transform (Pagel 1999). At lambda = 1 the tree is left unmodified. The tree approaches a star phylogeny as lambda approaches zero. lambda values larger than one are undefined Negative lambda values are not allowed.

Value

Rescaled phylogenetic tree.

Author(s)

Silvia Castiglione, Pasquale Raia

References

Castiglione, S., Serio, C., Piccolo, M., Mondanaro, A., Melchionna, M., Di Febbraro, M., Sansalone, G., Wroe, S., & Raia, P. (2020). The influence of domestication, insularity and sociality on the tempo and mode of brain size evolution in mammals. *Biological Journal of the Linnean Society*,132: 221-231. doi:10.1093/biolinnean/blaa186

Pagel, M. (1999). Inferring the historical patterns of biological evolution. *Nature*, 401:877-884.

Examples

```
## Not run:
ape::rtree(100)->tree
phytools::fastBM(tree)->y
max(diag(vcv(tree)))->H

RRphylo::RRphylo(tree,y)->RR
rescaleRR(tree,RR=RR)->treeRR

rescaleRR(tree,height=H/3)->tree_height

rescaleRR(tree,trend=5)->tree_trend

rescaleRR(tree,delta=0.5)->tree_delta05
rescaleRR(tree,delta=2)->tree_delta2

rescaleRR(tree,kappa=0.5)->tree_kappa

rescaleRR(tree,lambda=0.5)->tree_lambda

## End(Not run)
```

---

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"),csvfile=NULL)
```

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	has been deprecated; please see the argument csvfile instead.
csvfile	if results should be saved to a .csv file, a character indicating the name of the csv file and the path where it is to be saved. If no path is indicated the file is stored in the working directory. If left unspecified, no file will be saved.

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

```
## Not run:
    data("DataApes")
    DataApes$PCstage->PCstage
    DataApes$Tstage->Tstage

    cc<- 2/parallel::detectCores()
    RRphylo(tree=Tstage,y=PCstage,clus=cc)->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)
  # for a focal species
    retrieve.angles(evo.p,wishlist="anglesMRCA",random="no",focus="species",
    species="Sap")
  # for both focal node and species
    retrieve.angles(evo.p,wishlist="anglesMRCA",random="no",focus="both",
    node=68,species="Sap")
  # without any specific requirement
    retrieve.angles(evo.p,wishlist="anglesMRCA",random="no",focus="none")

 # 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)
  # for a focal species
    retrieve.angles(evo.p,wishlist="angleDir",random="no",focus="species",
    species="Sap")
  # for both focal node and species
    retrieve.angles(evo.p,wishlist="angleDir",random="no",focus="both",
    node=68,species="Sap")
  # without any specific requirement
    retrieve.angles(evo.p,wishlist="angleDir",random="no",focus="none")

 # 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)
  # for a focal species
    retrieve.angles(evo.p,wishlist="angles.between.species",random="no",
    focus="species", species="Sap")
```

```
  # for both focal node and species
    retrieve.angles(evo.p,wishlist="angles.between.species",random="no",
    focus="both",node=68,species="Sap")
  # without any specific requirement
    retrieve.angles(evo.p,wishlist="angles.between.species",random="no",
    focus="none")


# Case 2. "evo.dir" with performing randomization
    evo.dir(RR,angle.dimension="rates",pair.type="node",node=57,
    random="yes",nrep=10)->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)
  # for a focal species
    retrieve.angles(evo.p,wishlist="anglesMRCA",random="yes", focus="species",
    species="Sap")
  # for both focal node and species
    retrieve.angles(evo.p,wishlist="anglesMRCA",random="yes",focus="both",
    node=68,species="Sap")
  # without any specific requirement
    retrieve.angles(evo.p,wishlist="anglesMRCA",random="yes",focus="none")

 # 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)
  # for a focal species
    retrieve.angles(evo.p,wishlist="angleDir",random="yes",focus="species",
    species="Sap")
  # for both focal node and species
    retrieve.angles(evo.p,wishlist="angleDir",random="yes",focus="both",
    node=68, species="Sap")
  # without any specific requirement
    retrieve.angles(evo.p,wishlist="angleDir",random="yes",focus="none")

 # 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)
  # for a focal species
    retrieve.angles(evo.p,wishlist="angles.between.species",random="yes",
    focus="species", species="Sap")
  # for both focal node and species
    retrieve.angles(evo.p,wishlist="angles.between.species",random="yes",
    focus="both",node=68,species="Sap")
  # without any specific requirement
    retrieve.angles(evo.p,wishlist="angles.between.species",random="yes",
    focus="none")

## End(Not run)
```

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, the vector of predicted phenotypes, 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,rootV=NULL,aces=NULL,x1=NULL,aces.x1=NULL,clus=0.5)
```

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. In any case, y must be named. In case of categorical variable, this should be supplied to the function as a numeric vector.
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. See RRphylo vignette - covariate for details.
rootV	phenotypic value (values if multivariate) at the tree root. If rootV=NULL the function 'learns' about the root value from the 10% tips being closest in time to the tree root, weighted by their temporal distance from the root itself (close tips phenotypes weigh more than more distant tips).
aces	a named vector (or matrix if y is multivariate) of ancestral character values at nodes. Names correspond to the nodes in the tree. See RRphylo vignette - aces for details.
x1	the additional predictor(s) to be indicated to perform the multiple version of RRphylo. 'x1' vector/matrix 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 predictors (separately for each predictor) as well, and taking the vector of ancestral states and tip values to form the x1. See RRphylo vignette - predictor for details.
aces.x1	a named vector/matrix of ancestral character values at nodes for x1. It must be indicated if both aces and x1 are specified. Names/rownames correspond to the nodes in the tree.
clus	the proportion of clusters to be used in parallel computing (only if y is multivariate). To run the single-threaded version of RRphylo set clus = 0.

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**. Note, tip labels are ordered according to their position in the tree.
>
> **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. It is a matrix in the case of multivariate data.
>
> **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` by the inner function `optL`. With multivariate data, several `optL` runs are performed. Hence, the function provides a single lambda for each individual variable.
>
> **ace.values** if `aces` are specified, the function returns a dataframe containing the corresponding node number on the RRphylo tree for each node , along with estimated values.
>
> **x1.rate** if `x1` is specified, the function returns the partial regression coefficient for `x1`.

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*, 9: 974-983.doi:10.1111/2041-210X.12954

Serio, C., Castiglione, S., Tesone, G., Piccolo, M., Melchionna, M., Mondanaro, A., Di Febbraro, M., & Raia, P.(2019). Macroevolution of toothed whales exceptional relative brain size. *Evolutionary Biology*, 46: 332-342. doi:10.1007/s11692-019-09485-7

Melchionna, M., Mondanaro, A., Serio, C., Castiglione, S., Di Febbraro, M., Rook, L.,Diniz-Filho,J.A.F., Manzi, G., Profico, A., Sansalone, G., & Raia, P.(2020).Macroevolutionary trends of brain mass in Primates. *Biological Journal of the Linnean Society*, 129: 14-25. doi:10.1093/biolinnean/blz161

Castiglione, S., Serio, C., Mondanaro, A., Melchionna, M., Carotenuto, F., Di Febbraro, M., Profico, A., Tamagnini, D., & Raia, P. (2020). Ancestral State Estimation with Phylogenetic Ridge Regression. *Evolutionary Biology*, 47: 220-232. doi:10.1007/s11692-020-09505-x

Castiglione, S., Serio, C., Piccolo, M., Mondanaro, A., Melchionna, M., Di Febbraro, M., Sansalone, G., Wroe, S., & Raia, P. (2020). The influence of domestication, insularity and sociality

on the tempo and mode of brain size evolution in mammals. *Biological Journal of the Linnean Society*,132: 221-231. doi:10.1093/biolinnean/blaa186

Examples

```
 ## Not run:
data("DataOrnithodirans")
DataOrnithodirans$treedino->treedino
DataOrnithodirans$massdino->massdino

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

# Case 2. "RRphylo" accounting for the effect of a covariate
# "RRphylo" on the covariate in order to retrieve ancestral state values
c(RRcova$aces,massdino)->cov.values
c(rownames(RRcova$aces),names(massdino))->names(cov.values)

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

# Case 3. "RRphylo" specifying the ancestral states
data("DataCetaceans")
DataCetaceans$treecet->treecet
DataCetaceans$masscet->masscet
DataCetaceans$brainmasscet->brainmasscet
DataCetaceans$aceMyst->aceMyst

RRphylo(tree=treecet,y=masscet,aces=aceMyst)->RR

# Case 4. Multiple "RRphylo"
library(ape)
drop.tip(treecet,treecet$tip.label[-match(names(brainmasscet),treecet$tip.label)])->treecet.multi
masscet[match(treecet.multi$tip.label,names(masscet))]->masscet.multi

RRphylo(tree=treecet.multi, y=masscet.multi)->RRmass.multi
RRmass.multi$aces[,1]->acemass.multi
c(acemass.multi,masscet.multi)->x1.mass

RRphylo(tree=treecet.multi,y=brainmasscet,x1=x1.mass)->RR

# Case 5. Categorical and multiple "RRphylo" with 2 additional predictors
library(phytools)

set.seed(1458)
rtree(50)->tree
fastBM(tree)->y
jitter(y)*10->y1
rep(1,length(y))->y2
y2[sample(1:50,20)]<-2
names(y2)<-names(y)

c(RRphylo(tree,y1)$aces[,1],y1)->x1
```

```
RRphylo(tree,y2)->RRcat ### this is the RRphylo on the categorical variable
c(RRcat$aces[,1],y2)->x2

cbind(c(jitter(mean(y1[tips(tree,83)]))),1),
      c(jitter(mean(y1[tips(tree,53)]))),2))->acex
c(jitter(mean(y[tips(tree,83)])),jitter(mean(y[tips(tree,53)])))->acesy
names(acesy)<-rownames(acex)<-c(83,53)

RRphylo(tree,y,aces=acesy,x1=cbind(x1,x2),aces.x1 = acex)


## End(Not run)
```

RRphylo-defunct   *Defunct functions in Package RRphylo*

Description

These functions are no longer available.

- swap.phylo: This function is defunct. Please check [overfitRR](), instead.

Usage

```
swap.phylo()
```

RRphylo-deprecated   *Deprecated functions in Package RRphylo*

Description

These functions still work but will be removed (defunct) in the next version.

---

scaleTree   *Phylogenetic tree calibration*

Description

The function is a wrapper around the functions "scalePhylo", "assign.ages", and "assign.brlen" written by Gene Hunt (http://paleobiology.si.edu/staff/individuals/hunt.cfm). It rescales tree branch lengths according to given calibration dates.

Usage

```
scaleTree(tree, tip.ages=NULL, node.ages=NULL,min.branch=NULL)
```

Arguments

tree	a phylogenetic tree. The tree needs not to be ultrametric and fully dichotomous.
tip.ages	a named vector including the ages (i.e. distance from the youngest tip within the tree) of the tips to be changed. If unspecified, the function assumes all the tips are correctly placed with respect to the root.
node.ages	a named vector including the ages (i.e. distance from the youngest tip within the tree) of the nodes to be changed. If no calibration date for nodes is supplied, the tree root is fixed and the function shifts node position only where needed to fit tip ages.
min.branch	has been deprecated.

Value

Rescaled phylogentic tree with tip labels ordered according to their position in the tree.

Author(s)

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

See Also

scaleTree vignette

Examples

```
library(ape)
library(phytools)

data("DataFelids")
DataFelids$treefel->tree

max(nodeHeights(tree))->H

#### Example 1 ####
rep(0,4)->tipAges
names(tipAges)<-tips(tree,146)
scaleTree(tree,tipAges)->treeS1

edge.col<-rep("black",nrow(tree$edge))
edge.col[which(treeS1$edge[,2]%in%getDescendants(treeS1,146))]<-"red"

layout(2:1)
plot(tree,edge.color = edge.col,show.tip.label=FALSE)
plot(treeS1,edge.color = edge.col,show.tip.label=FALSE)

#### Example 2 ####
nodeAges<-c(23.5,15.6)
names(nodeAges)<-c(85,139)
```

```
scaleTree(tree,node.ages=nodeAges)->treeS2

edge.col<-rep("black",nrow(tree$edge))
edge.col[which(treeS1$edge[,2]%in%c(getDescendants(treeS1,85),
                                    getDescendants(treeS1,139)))]<-"red"

layout(2:1)
plot(tree,edge.color = edge.col,show.tip.label=FALSE)
nodelabels(bg="w",frame="n",node=c(85,139),col="green")
plot(treeS2,edge.color = edge.col,show.tip.label=FALSE)
nodelabels(bg="w",frame="n",node=c(85,139),col="green")

#### Example 3 ####
16->nodeAges
names(nodeAges)<-"145"
tipAges<-19
names(tipAges)<-tree$tip.label[1]
scaleTree(tree,tip.ages = tipAges,node.ages=nodeAges)->treeS3

edge.col<-rep("black",nrow(tree$edge))
edge.col[which(treeS3$edge[,2]%in%c(1,getMommy(tree,1),
                                    getDescendants(treeS3,145)))]<-"red"

layout(2:1)
plot(tree,edge.color = edge.col,show.tip.label=FALSE)
nodelabels(bg="w",frame="n",node=145,col="green")
plot(treeS3,edge.color = edge.col,show.tip.label=FALSE)
nodelabels(bg="w",frame="n",node=145,col="green")
```

---

search.conv   *Searching for morphological convergence among species and clades*

Description

The function scans a phylogenetic tree looking for morphological convergence between entire clades or species evolving under specific states.

Usage

```
search.conv(RR=NULL,tree=NULL,y,nodes=NULL,state=NULL,aceV=NULL,
  min.dim=NULL,max.dim=NULL,min.dist=NULL,declust=FALSE,nsim=1000,rsim=1000,
  clus=.5,filename=NULL)
```

Arguments

RR	an object produced by [RRphylo](). This is not indicated if convergence among states is tested.
tree	a phylogenetic tree. The tree needs not to be ultrametric or fully dichotomous. This is not indicated if convergence among clades is tested.

y	a multivariate phenotype. The object y should be either a matrix or dataframe with species names as rownames.
nodes	node pair to be tested. If unspecified, the function automatically searches for convergence among clades. Notice the node number must refer to the dichotomic version of the original tree, as produced by RRphylo.
state	the named vector of tip states. The function tests for convergence within a single state or among different states (this latter case is especially meant to test for iterative evolution as for example the appearance of repeated morphotypes into different clades). In both cases, the state for non-focal species (i.e. not belonging to any convergent group) must be indicated as "nostate".
aceV	phenotypic values at internal nodes. The object aceV should be either a matrix or dataframe with nodes (referred to the dichotomic version of the original tree, as produced by RRphylo) as rownames. If aceV are not indicated, ancestral phenotypes are estimated via RRphylo.
min.dim	the minimum size of the clades to be compared. When nodes is indicated, it is the minimum size of the smallest clades in nodes, otherwise it is set at one tenth of the tree size.
max.dim	the maximum size of the clades to be compared. When nodes is indicated, it is min.dim*2 if the largest clade in nodes is smaller than this value, otherwise it corresponds to the size of the largest clade. Without nodes it is set at one third of the tree size.
min.dist	the minimum distance between the clades to be compared. When nodes is indicated, it is the distance between the pair. Under the automatic mode, the user can choose whether time distance or node distance (i.e. the number of nodes intervening between the pair) should be used. If time distance has to be considered, min.dist should be a character argument containing the word "time" and then the actual time distance to be used. The same is true for node distance, but the word "node" must precede the node distance to be used. For example, if the user want to test only clades more distant than 10 time units, the argument should be "time10". If clades separated by more than 8 nodes has to be tested, the argument min.dist should be "node8". If left unspecified, it automatically searches for convergence between clades separated by a number of nodes bigger than one tenth of the tree size.
declust	if species under a given state (or a pair of states) to be tested for convergence are phylogenetically closer than expected by chance, trait similarity might depend on proximity rather than true convergence. In this case, by setting declust = TRUE, tips under the focal state (or states) are removed randomly until clustering disappears. A minimum of 3 species per state is enforced to remain anyway.
nsim	number of simulations to perform sampling within the theta random distribution. It is set at 1000 by default.
rsim	number of simulations to be performed to produce the random distribution of theta values. It is set at 1000 by default.
clus	the proportion of clusters to be used in parallel computing. To run the single-threaded version of search.conv set clus = 0.
filename	is deprecated. search.conv does not return plots anymore, check the function [plotConv](#) instead.

Value

If convergence between clades is tested, the function returns a list including:

- **$node pairs**: a dataframe containing for each pair of nodes:
  - ang.bydist.tip: the mean theta angle between clades divided by the time distance.
  - ang.conv: the mean theta angle between clades plus the angle between aces, divided by the time distance.
  - ang.ace: the angle between aces.
  - ang.tip: the mean theta angle between clades.
  - nod.dist: the distance intervening between clades in terms of number of nodes.
  - time.dist: the time distance intervening between the clades.
  - p.ang.bydist: the p-value computed for ang.bydist.tip.
  - p.ang.conv: the p-value computed for ang.conv.
  - clade.size: the size of clades.
- **$node pairs comparison**: pairwise comparison between significantly convergent pairs (all pairs if no instance of significance was found) performed on the distance from group centroids (the mean phenotype per clade).
- **$average distance from group centroids**: smaller average distances mean less variable phenotypes within the pair.

If convergence between (or within a single state) states is tested, the function returns a list including:

- **state.res** a dataframe including for each pair of states (or single state):
  - ang.state: the mean theta angle between species belonging to different states (or within a single state).
  - ang.state.time: the mean of theta angle between species belonging to different states (or within a single state) divided by time distance.
  - p.ang.state: the p-value computed for ang.state.
  - p.ang.state.time: the p-value computed for ang.state.time.

- **plotData** a dataframe including data to plot the results via `plotConv`

.

Author(s)

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

References

Castiglione, S., Serio, C., Tamagnini, D., Melchionna, M., Mondanaro, A., Di Febbraro, M., Profico, A., Piras, P.,Barattolo, F., & Raia, P. (2019). A new, fast method to search for morphological convergence with shape data. *PLoS ONE*, 14, e0226949. https://doi.org/10.1371/journal.pone.0226949

See Also

search.conv vignette

plotConv vignette

Examples

```
## Not run:
data("DataFelids")
DataFelids$PCscoresfel->PCscoresfel
DataFelids$treefel->treefel
DataFelids$statefel->statefel
cc<- 2/parallel::detectCores()

RRphylo(treefel,PCscoresfel,clus=cc)->RRfel


## Case 1. searching convergence between clades
# by setting min.dist as node distance
search.conv(RR=RRfel, y=PCscoresfel, min.dim=5, min.dist="node9",clus=cc)
# by setting min.dist as time distance
search.conv(RR=RRfel, y=PCscoresfel, min.dim=5, min.dist="time38",clus=cc)

## Case 2. searching convergence within a single state
search.conv(tree=treefel, y=PCscoresfel, state=statefel,declust=TRUE,clus=cc)

## End(Not run)
```

---

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"),node = NULL, state
  = NULL, cov = NULL, nrep = 1000, f = NULL,filename=NULL)
```

Arguments

RR   an object fitted by the function RRphylo.

status.type   whether the "clade" or "sparse" condition must be tested.

node   under the "clade" condition, the node (clade) to be tested for the rate shift.
                When multiple nodes are tested, they need to be written as in the example be-
                low. If node is left unspecified, the function performs under the 'auto-recognize'
                feature, meaning it will automatically test individual clades for deviation of their
                rates from the background rate of the rest of the tree (see details).

state   the state of the tips specified under the "sparse" condition.

cov   the covariate vector 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.

filename   is deprecated. search.shift does not return plots anymore, check the function
                [plotShift](#) instead.

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. Un-
der 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. 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, multi-
ple 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. 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 `"clade"` case without specifying nodes (i.e. 'auto-recognize') a list including:

$all.clades for each detected node, the data-frame includes 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) and the probability that it do represent a real shift. Probabilities are contrasted to simulations shuffling the rates across the tree branches for a number of replicates 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.

$single.clades the same as with 'all.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.

Under `"clade"` condition by specifying the `node` argument:

$all.clades.together if more than one node is tested, this specifies the average rate difference and the significance of the rate shift, by considering all the specified nodes as 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.

$single.clades this gives the significance for individual clades, tested separately. As previously, 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:

$state.results for each state, the data-frame includes the average rate difference (computed as the mean rate over all leaves evolving under a given state, minus the average rate for each other state or the rest of the tree) and the probability that the shift is real. Large rates are significantly larger (at alpha = 0.05), when the probability is > 0.975; and small rates are significantly small for p < 0.025. States are compared 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*, 9: 974-983.doi:10.1111/2041-210X.12954

See Also

search.shift vignette

plotShift vignette

Examples

```
## Not run:
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(RR=dinoRates,status.type="clade")
# testing two hypothetical clades
search.shift(RR=dinoRates,status.type="clade",node=c(696,746))

# Case 1.2 "sparse" condition
# testing the sparse condition.
search.shift(RR=dinoRates,status.type= "sparse",state=statedino)


# Case 2. Accounting for the effect of a covariate

# Case 2.1 "clade" condition
search.shift(RR=dinoRates,status.type= "clade",cov=massdino)

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

## End(Not run)
```

---

search.trend | *Searching for evolutionary trends in phenotypes and rates*

Description

This function searches for evolutionary trends in the phenotypic mean and the evolutionary rates for the entire tree and individual clades.

Usage

```
search.trend(RR,y,x1=NULL,x1.residuals = FALSE,
 node=NULL,cov=NULL,nsim=100,clus=0.5,ConfInt=NULL,filename=NULL)
```

Arguments

RR	an object produced by [RRphylo]().
y	the named vector (or matrix if multivariate) of phenotypes.
x1	the additional predictor to be specified if the RR object has been created using an additional predictor (i.e. multiple version of RRphylo). 'x1' vector 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 predictor as well, and taking the vector of ancestral states and tip values to form the x1. Note: only one predictor at once can be specified.
x1.residuals	logical specifying whether the residuals of regression between y and x1 should be inspected for a phenotypic trend (see details and examples below). Default is FALSE.
node	the node number of individual clades to be specifically tested and contrasted to each other. It is NULL by default. Notice the node number must refer to the dichotomic version of the original tree, as produced by RRphylo.
cov	the covariate values to be specified if the RR object has been created using a covariate for rates calculation. As for RRphylo, '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 (see the example below).
nsim	number of simulations to be performed. It is set at 100 by default.
clus	the proportion of clusters to be used in parallel computing. To run the single-threaded version of search.trend set clus = 0.
ConfInt	is deprecated.
filename	is deprecated. search.trend does not return plots anymore, check the function [plotTrend]() instead.

Details

The function simultaneously returns the regression of phenotypes and phenotypic evolutionary rates against age tested against Brownian motion simulations to assess significance. To this aim rates are rescaled in the 0-1 range and then logged. To assess significance, slopes are compared to a family of simulated slopes (BMslopes, where the number of simulations is equal to nsim), generated under the Brownian motion, using the fastBM function in the package **phytools**. Individual nodes are compared to the rest of the tree in different ways depending on whether phenotypes or rates (always unscaled in this case) versus age regressions are tested. With the former, the regression slopes for individual clades and the slope difference between clades is contrasted to slopes obtained through Brownian motion simulations. For the latter, regression models are tested and contrasted to each other referring to estimated marginal means, by using the emmeans function in the package **emmeans**.

The [multiple regression version of RRphylo]() allows to incorporate the effect of an additional predictor in the computation of evolutionary rates without altering the ancestral character estimation. Thus, when a multiple RRphylo output is fed to search.trend, the predictor effect is accounted for on the absolute evolutionary rates, but not on the phenotype. However, in some situations the user might want to factor out the predictor effect on phenotypes as well. Under the latter circumstance,

by setting the argument `x1.residuals = TRUE`, the residuals of the response to predictor regression are used as to represent the phenotype.

Value

The function returns a list object containing:

$trends.data a 'RRphyloList' object including:

1. `$phenotypeVStime`: a data frame of phenotypic values (or y versus x1 regression residuals if `x1.residuals=TRUE`) and their distance from the tree root for each node (i.e. ancestral states) and tip of the tree.
2. `$absrateVStime`: a data frame of RRphylo rates and the distance from the tree root (age). If y is multivariate, it also includes the multiple rates for each y vector. If node is specified, each branch is classified as belonging to an indicated clade.
3. `$rescaledrateVStime`: a data frame of rescaled RRphylo rates and the distance from the tree root (age). If y is multivariate, it also includes the multiple rates for each y vector. If node is specified, each branch is classified as belonging to an indicated clade. NAs correspond either to very small values or to outliers which are excluded from the analysis.

$phenotypic.regression results of phenotype (y versus x1 regression residuals) versus age regression. It reports a p-value for the regression slope between the variables (p.real), a p-value computed contrasting the real slope to Brownian motion simulations (p.random), and a parameter indicating the deviation of the phenotypic mean from the root value in terms of the number of standard deviations of the trait distribution (dev). dev is 0 under Brownian Motion. Only p.random should be inspected to assess significance.

$rate.regression results of the rates (rescaled absolute values) versus age regression. It reports a p-value for the regression between the variables (p.real), a p-value computed contrasting the real slope to Brownian motion simulations (p.random), and a parameter indicating the ratio between the range of phenotypic values and the range of such values halfway along the tree height, divided to the same figure under Brownian motion (spread). spread is 1 under Brownian Motion. Only p.random should be inspected to assess significance.

$ConfInts a 'RRphyloList' object including the 95% confidence intervals around regression slopes of phenotypes and rates (both rescaled and unscaled absolute rates) produced according to the Brownian motion model of evolution.

If specified, individual nodes are tested as the whole tree, the results are summarized in the objects:

$node.phenotypic.regression results of phenotype (or y versus x1 regression residuals) versus age regression through node. It reports the slope for the regression between the variables at node (slope), a p-value computed contrasting the real slope to Brownian motion simulations (p.random), the difference between estimated marginal means predictions for the group and for the rest of the tree (emm.difference), and a p-value for the emm.difference (p.emm).

$node.rate.regression results of the rates (absolute values) versus age regression through node. It reports the difference between estimated marginal means predictions for the group and for the rest of the tree (emm.difference), a p-value for the emm.difference (p.emm), the regression slopes for the group (slope.node) and for the rest of the tree (slope.others), and a p-value for the difference between such slopes (p.slope).

If more than one node is specified, the object **$group.comparison** reports the same results as $node.phenotypic.regression and $node.rate.regression obtained by comparing individual clades to each other.

Author(s)

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

References

Castiglione, S., Serio, C., Mondanaro, A., Di Febbraro, M., Profico, A., Girardi, G., & Raia, P. (2019) Simultaneous detection of macroevolutionary patterns in phenotypic means and rate of change with and within phylogenetic trees including extinct species. *PLoS ONE*, 14: e0210101. https://doi.org/10.1371/journal.pone.0210101

See Also

search.trend vignette

plotTrend vignette

plotTrend

Examples

```
 ## Not run:
data("DataOrnithodirans")
DataOrnithodirans$treedino->treedino
DataOrnithodirans$massdino->massdino
cc<- 2/parallel::detectCores()

# Extract Pterosaurs tree and data
library(ape)
extract.clade(treedino,746)->treeptero
massdino[match(treeptero$tip.label,names(massdino))]->massptero
massptero[match(treeptero$tip.label,names(massptero))]->massptero

# Case 1. "RRphylo" whitout accounting for the effect of a covariate
RRphylo(tree=treeptero,y=log(massptero))->RRptero

# Case 1.1. "search.trend" whitout indicating nodes to be tested for trends
search.trend(RR=RRptero, y=log(massptero), nsim=100, clus=cc,cov=NULL,node=NULL)

# Case 1.2. "search.trend" indicating nodes to be specifically tested for trends
search.trend(RR=RRptero, y=log(massptero), nsim=100, node=143, clus=cc,cov=NULL)


# Case 2. "RRphylo" accounting for the effect of a covariate
# "RRphylo" on the covariate in order to retrieve ancestral state values
RRphylo(tree=treeptero,y=log(massptero))->RRptero
c(RRptero$aces,log(massptero))->cov.values
names(cov.values)<-c(rownames(RRptero$aces),names(massptero))
RRphylo(tree=treeptero,y=log(massptero),cov=cov.values)->RRpteroCov

# Case 2.1. "search.trend" whitout indicating nodes to be tested for trends
search.trend(RR=RRpteroCov, y=log(massptero), nsim=100, clus=cc,cov=cov.values)
```

```
# Case 2.2. "search.trend" indicating nodes to be specifically tested for trends
search.trend(RR=RRpteroCov, y=log(massptero), nsim=100, node=143, clus=cc,cov=cov.values)


# Case 3. "search.trend" on multiple "RRphylo"
data("DataCetaceans")
DataCetaceans$treecet->treecet
DataCetaceans$masscet->masscet
DataCetaceans$brainmasscet->brainmasscet
DataCetaceans$aceMyst->aceMyst

drop.tip(treecet,treecet$tip.label[-match(names(brainmasscet),treecet$tip.label)])->treecet.multi
masscet[match(treecet.multi$tip.label,names(masscet))]->masscet.multi

RRphylo(tree=treecet.multi,y=masscet.multi)->RRmass.multi
RRmass.multi$aces[,1]->acemass.multi
c(acemass.multi,masscet.multi)->x1.mass

RRphylo(tree=treecet.multi,y=brainmasscet,x1=x1.mass)->RRmulti

# incorporating the effect of body size at inspecting trends in absolute evolutionary rates
search.trend(RR=RRmulti, y=brainmasscet,x1=x1.mass,clus=cc)

# incorporating the effect of body size at inspecting trends in both absolute evolutionary
# rates and phenotypic values (by using brain versus body mass regression residuals)
search.trend(RR=RRmulti, y=brainmasscet,x1=x1.mass,x1.residuals=TRUE,clus=cc)

## End(Not run)
```

---

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"), trend.type = c("linear", "stepwise"),tr = 10, t.shift = 0.5,
  es=2, ds=1)
```

Arguments

tree	a phylogenetic tree.
nY	the number of phenotypes to simulate.
s2	value of the Brownian 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 phenotypic mean or in the rate of evolution. A variation in
                the phenotypic mean over time (a phenotypic trend) is obtained by selecting the
                option "drift". A trend in the rate of evolution produces an increased variance
                in the residuals over time. This is obtained by specifying the option "trend".

trend.type   two kinds of heteroscedastic residuals are generated under the "trend" type.
                The option "linear" produces an exponential linear increase (or decrease) in
                heteroscedasticity, whereas the "stepwise" option produces an increase (or de-
                crease) after a specified point in time.

tr   the intensity of the trend 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 change in the
                rate of evolution is indicated to apply.

es   when trend.type="linear", es is a scalar representing the exponent at which
                the evolutionary time (i.e. distance from the root) scales to change to phenotypic
                variance. With es = 1 the phenotypic rate will be trendless, with es < 1 the
                variance of the phenotypes will decrease exponentially towards the present and
                the other way around with es > 1.

ds   a scalar indicating the change in phenotypic mean in the unit time, in type="drift"
                case. With ds = 0 the phenotype will be trendless, with ds < 0 the phenotypic
                mean will decrease exponentially towards the present and the other way around
                with ds > 0.

Details

Note that setBM differs from fastBM in that the produced phenotypes are checked for the existence
of a temporal trend in the phenotype. The user may specify whether she wants trendless data
(option "brown"), phenotypes trending in time (option "drift"), or phenotypes whose variance
increases/decreases exponentially over time, consistently with the existence of a trend in the rate of
evolution (option "trend"). In the latter case, the user may indicate the intensity of the trend (by
applying different values of es), and whether it should occur after a given proportion of the tree
height (hence a given point back in time, specified by the argument t.shift). 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 * 10e-3,in order to make it non-ultrametric.

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", ds=2)
setBM(tree=treedino, nY= 1, type="trend", trend.type="linear", es=2)
```

---

sig2BM	*Brownian Motion rate computation*

Description

The function computes rate of phenotypic evolution along a phylogeny assuming Brownian Motion model of evolution.

Usage

```
sig2BM(tree,y)
```

Arguments

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

y   either a single vector variable or a multivariate dataset. In any case, y must be named.

Value

The Brownian Motion rate of phenotypic evolution for each variable in y.

Author(s)

Pasquale Raia, Silvia Castiglione

Examples

```
### Univariate data ###
data(DataCetaceans)
DataCetaceans$treecet->treecet
DataCetaceans$masscet->masscet
sig2BM(tree=treecet,y=masscet)

### Multivariate data ###
data(DataUng)
DataUng$treeung->treeung
DataUng$PCscoresung->PCscores
sig2BM(tree=treeung,y=PCscores)
```

---

sizedsubtree   *Find a node subtending to a clade of desired 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)
```

StableTraitsR   *Run StableTraits from within R*

Description

This function runs StableTraits and StableTraitsSum (*Elliot and Mooers 2014*) from within the R environment and returns its output into the workspace.

Usage

```
StableTraitsR(tree,y,path,output=NULL,aces=NULL,argST=NULL,argSTS=NULL)
```

Arguments

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

y   a named vector of phenotypic trait.

path   the folder path where the StableTraits output will be stored. Notice that the input tree and data (modified automatically if the original tree is not fully dichotomous or if aces are specified) will be stored in this folder as well.

output   name of the output to be returned, if unspecified it will be named "output".

aces   a named vector of ancestral character values at nodes specified in advance. Names correspond to the nodes in the tree.

argST   a list of further arguments passed to StableTraits. If the argument has no value (for example "brownian") it must be specified as TRUE.

argSTS   list of further arguments passed to StableTraitsSum. If the argument has no value (for example "brownian") it must be specified as TRUE.

Details

The StableTraits software is available at https://mickelliot.com/, along with instructions for compilation. Once it is installed, the user must set as R working directory the folder where the StableTraits software are installed. Further information about the arguments and outputs of StableTraits and StableTraitsSum can be found at https://mickelliot.com/. StableTraitsR automatically recognizes which Operating System is running on the computer (it has been tested successfully on MacOS and Windows machines).

Value

The function returns a 'list' containing the output of StableTraits and StableTraitsSum.

$progress a table reporting the DIC and PRSF diagnostics.

$rates_tree a copy of the original tree with branch lengths set to the evolutionary rate imputed by the stable reconstruction. Specifically, each branch length is equal to the absolute difference in the stable reconstruction occurring on that branch divided by the square root of the input branch length.

$rates the original branch lengths, evolutionary rates, node height and (optionally) scaled branch lengths.

$aces the median estimates of ancestral states and stable parameters along with the 95% credible interval.

$brownian_tree if "brownian" is TRUE in argSTS, a copy of the original tree with branch lengths set such that the Brownian motion reconstruction of the character on this tree is approximately the same as the stable ancestral reconstruction.

$ace.prior.values if aces is specified, the function returns a dataframe containing the corresponding node number on the RRphylo tree for each node, the original (preset) and the estimated values, and the 95% credible interval.

Author(s)

Silvia Castiglione, Carmela Serio, Pasquale Raia

References

Elliot, M. G., & Mooers, A. Ø. (2014). Inferring ancestral states without assuming neutrality or gradualism using a stable model of continuous character evolution. *BMC evolutionary biology*, 14: 226. doi.org/10.1186/s12862-014-0226-8

Examples

```
## Not run:
library(ape)
library(phytools)

# Set as working directory the folder where StableTraits software are stored
# setwd("~/StableTraits")

dir.create("Analyses")
rtree(100)->tree
fastBM(tree)->y
c(1,2,3)->acev
sample(Ntip(tree)+seq(1:Nnode(tree)),3)->names(acev)
StableTraitsR(tree,y,path="Analyses/",output="my_output",aces=acev,
argST=list(iterations=500000,chains=4),argSTS=list(brownian=TRUE))->ST

## End(Not run)
```

---

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,node=NULL,si=0.5,si2=0.5,plot.swap=FALSE)
```

Arguments

tree	a phylogenetic tree. The tree needs not to be ultrametric or fully dichotomous.
node	if specified, the clades subtended by such node(s) are imposed to be monophyletic. In this case, the function can still swap tips *within* the clade.
si	the proportion of tips whose topologic arrangement will be swapped.
si2	the proportion of nodes whose age will be changed.
plot.swap	if TRUE, the function plots the swapped tree. Swapped positions appear in red. Nodes with altered ages appear in green.

Details

swapONE 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. Note, tip labels are ordered according to their position in the tree.

Author(s)

Silvia Castiglione, Pasquale Raia, 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

```
## Not run:
data("DataOrnithodirans")
DataOrnithodirans$treedino->treedino

## Case 1. change the topology and the branch lengths for the entire tree
swapONE(tree=treedino,si=0.5,si2=0.5,plot.swap=FALSE)

## Case 2. change the topology and the branch lengths of the
##         tree by keeping the monophyly of a specific clade
swapONE(tree=treedino,node=422,si=0.5,si2=0.5,plot.swap=FALSE)

## End(Not run)
```

---

tips   *Get descending tips*

Description

The function returns the numbers or labels of tips descending from a given node.

Usage

```
tips(tree,node,labels=TRUE)
```

Arguments

tree	a phylogenetic tree. The tree needs not to be ultrametric and fully dichotomous.
node	the number of focal node
labels	if TRUE (default) the function returns the labels of descending tips.

Value

The tips, either labels or numbers depending on the argument labels, descending from the node.

Author(s)

Silvia Castiglione, Pasquale Raia, Carmela Serio

Examples

```
data(DataOrnithodirans)
DataOrnithodirans$treedino->treedino
tips(tree=treedino,node=677,labels=FALSE)
tips(tree=treedino,node=677,labels=TRUE)
```

---

tree.merger   *Fast addition of tips and clades on an existing tree*

Description

The function attaches new tips and/or clades derived from a source phylogeny to a pre-existing backbone tree.

Usage

```
tree.merger(backbone,data,source.tree=NULL,age.offset=NULL,tip.ages =
  NULL, node.ages = NULL,min.branch=NULL,plot=TRUE,filename=NULL)
```

Arguments

backbone   the backbone tree to attach tips/clades on.

data   a dataset including as columns:

    1. bind = the tips/clades to be attached;
    2. reference = the reference tip or clade where 'bind' must be attached;
    3. poly = logical specifying if 'bind' and 'reference' should form a polyto-
       mous clade.

    See details for further explanations.

source.tree   the tree where 'bind' clades are to be extracted from. If no clade has to be
                attached, it can be left unspecified.

age.offset   if the most recent age (i.e. the maximum distance from the tree root) differs
                between the source and the backbone trees, the "age.offset" is the difference
                between them in this exact order (source minus backbone). It is positive when
                the backbone tree attains younger age than the source tree, and vice-versa.

tip.ages   as in [scaleTree](#), a named vector including the ages (i.e. the time distance from
                the youngest tip within the tree) of the tips. If unspecified, the function assumes
                all the tips on the backbone tree are correctly placed and places all the new tips at
                the maximum distance from the tree root (i.e. the present if the tips are extant).

node.ages   as in scaleTree, a named vector including the ages (i.e. the time distance from
                the youngest tip within the tree) of the nodes. The nodes must be defined by
                collating the names of the two phylogenetically furthest tips it subtends to, sep-
                arated by a "-" (see examples). If no calibration date for nodes is supplied, the
                function may shift the node position back in time as to place new tips/clades and
                to fit tip ages.

min.branch   has been deprecated.

plot   if TRUE, the function produces an interactive plotting device to check the placing
                of each bind.

filename   if plot=TRUE and provided a filename (with or without the path), the function
                stores a pdf file showing the plot of the entire phylogeny.

Details

The function attaches tips and/or clades from the source tree to the backbone tree according to
the data object. Within the latter, a clade, either to be bound or to be the reference, must be
indicated by collating the names of the two phylogenetically furthest tips belonging to it, separated
by a "-". Duplicated 'bind' produce error. Tips/clades set to be attached to the same 'reference'
are considered to represent a polytomy. Tips set as 'bind' which are already on the backbone tree
are removed from the latter and placed according to the 'reference'. See examples and vignette for
clarifications.

Value

Merged phylogenetic tree.

Author(s)

Silvia Castiglione, Carmela Serio, Pasquale Raia

References

Castiglione, S., Serio, C., Mondanaro, A., Melchionna, M., & Raia, P. (2022). Fast production of
large, time-calibrated, informal supertrees with tree.merger. *Palaeontology*, 65: e12588.https://doi.org/10.1111/pala.12588

See Also

tree.merger vignette; scaleTree vignette;

Examples

```
## Not run:
require(ape)
DataCetaceans$treecet->tree
data.frame(bind=c("Balaena_mysticetus-Caperea_marginata",
                  "Aetiocetus_weltoni",
                  "Saghacetus_osiris",
                  "Zygorhiza_kochii",
                  "Ambulocetus_natans",
                  "Kentriodon_pernix",
                  "Kentriodon_schneideri",
                  "Kentriodon_obscurus",
                  "Tursiops_truncatus-Delphinus_delphis",
                  "Kogia_sima",
                  "Grampus_griseus"),
           reference=c("Fucaia_buelli-Aetiocetus_weltoni",
                       "Aetiocetus_cotylalveus",
                       "Fucaia_buelli-Tursiops_truncatus",
                       "Saghacetus_osiris-Fucaia_buelli",
                       "Dalanistes_ahmedi-Fucaia_buelli",
                       "Kentriodon_schneideri",
                       "Phocoena_phocoena-Delphinus_delphis",
                       "Kentriodon_schneideri",
                       "Stenella_attenuata-Stenella_longirostris",
                       "Kogia_breviceps",
                       "Globicephala_melas-Pseudorca_crassidens"),
           poly=c(FALSE,
                  FALSE,
                  FALSE,
                  FALSE,
                  FALSE,
                  FALSE,
                  FALSE,
                  FALSE,
                  FALSE,
                  FALSE,
                  FALSE))->dato

c(Aetiocetus_weltoni=28.0,
```

```
    Saghacetus_osiris=33.9,
    Zygorhiza_kochii=34.0,
    Ambulocetus_natans=40.4,
    Kentriodon_pernix=15.9,
    Kentriodon_schneideri=11.61,
    Kentriodon_obscurus=13.65)->tip.ages
 c("Ambulocetus_natans-Fucaia_buelli"=52.6,
    "Balaena_mysticetus-Caperea_marginata"=21.5)->node.ages

 # remove some tips from the original tree and create a source tree
 drop.tip(tree,c(names(tip.ages),
                 tips(tree,131)[-which(tips(tree,131)%in%
                                 c("Caperea_marginata","Eubalaena_australis"))],
                 tips(tree,195)[-which(tips(tree,195)=="Tursiops_aduncus")]))->backtree
 drop.tip(tree,which(!tree$tip.label%in%c(names(tip.ages),
                                          tips(tree,131),
                                          tips(tree,195))))->sourcetree

 plot(backtree,cex=.6)
 plot(sourcetree,cex=.6)

 tree.merger(backbone=backtree,data=dato,source.tree=sourcetree,
             tip.ages=tip.ages,node.ages = node.ages, plot=TRUE)->treeM

## End(Not run)
```

---

treeCompare   *Visualize the difference between phylogenetic trees*

Description

The function scans a pair of phylogenetic trees to find topological differences.

Usage

```
treeCompare(tree, tree1, plot = TRUE)
```

Arguments

tree, tree1	a phylogenetic tree. The tree needs not to be ultrametric and fully dichotomous. Generic name and specific epithet must be separated by '_'.
plot	if TRUE, the function produces an interactive plotting device to check differences in species placement at the genus level.

Value

The function returns a data-frame indicating for each un-matching species its sister species/clades on both trees.

Author(s)

Silvia Castiglione, Carmela Serio, Antonella Esposito

Examples

```
## Not run:
DataSimians$tree->tree
set.seed(22)
swapONE(tree,si=0.5)[[1]]->tree1

treeCompare(tree,tree1)


## End(Not run)
```

---

treedataMatch   *Cross-reference tree and data*

Description

The function matches data names with tree tips. If either there is no data for a tip or it is not present
on the tree, the function removes the entry from both.

Usage

```
treedataMatch(tree,y)
```

Arguments

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

y   either a single vector variable or a multivariate dataset. In any case, y must be
                named.

Value

The function returns a list object. If no mismatch between tree and y is detected, the list only
includes the matrix of y ordered according to the order of tips on the tree ($y). If some tips on
the tree are missing from y, they are removed from the phylogeny. Thus, the list also includes
the pruned tree ($tree) and the vector of dropped tips ($removed.from.tree). Similarly, if some
entries in y are missing from the tree, the list also includes the vector of mismatching entry names
($removed.from.y). In this latter case, the first element of the list ($y) does not include the entries
$removed.from.y, so that it perfectly matches the phylogeny.

Author(s)

Silvia Castiglione, Pasquale Raia, Carmela Serio

Examples

```
data(DataCetaceans)
DataCetaceans$treecet->treecet
DataCetaceans$masscet->masscet
DataCetaceans$brainmasscet->brainmasscet

treedataMatch(tree=treecet,y=masscet)
treedataMatch(tree=treecet,y=brainmasscet)
```

Index