Package ‘apTreeshape’

October 12, 2022

Date 2018-03-22
Version 1.5-0.1
Title Analyses of Phylogenetic Treeshape
Author Nicolas Bortolussi, Michael Blum, Eric Durand, Olivier Francois, Odile Maliet
Maintainer Michael Blum <michael.blum@univ-grenoble-alpes.fr>
Depends R (>= 2.10), ape
Imports quantreg, cubature, coda, pbapply
Description Simulation and analysis of phylogenetic tree topologies using statistical indices. It is a companion library of the 'ape' package. It provides additional functions for reading, plotting, manipulating phylogenetic trees. It also offers convenient web-access to public databases, and enables testing null models of macroevolution using corrected test statistics. Trees of class ``phylo'' (from 'ape' package) can be converted easily. Implements methods described in Bortolussi et al. (2005) <doi:10.1093/bioinformatics/bti798> and Maliet et al. (2017) <doi:10.1101/224295>.
License GPL (>= 2)
Encoding latin1
Repository CRAN
NeedsCompilation no
Date/Publication 2020-10-12 14:12:28 UTC

R topics documented:

aldous.test ................................................................. 3
all.equal.treeshape ..................................................... 4
as.phylo.treeshape ..................................................... 6
as.treeshape ............................................................... 7
aux_lik ................................................................. 8
a_N ................................................................. 8
bind.trees .......................................................... 9
build.tree ......................................................... 10
carnivora.treeshape ........................................... 10
change_int .......................................................... 11
change_int_eta ..................................................... 11
cladsize ............................................................. 12
colless ............................................................... 13
cutreeshape ........................................................ 15
cytochrome ........................................................ 16
depth ................................................................. 16
enhance ............................................................... 17
enhance_etas ......................................................... 17
get_extinction_list ............................................... 18
get_PD_sample ..................................................... 19
get_tree_beta ....................................................... 20
hivtree.treeshape ............................................... 22
index.test .......................................................... 22
insert ................................................................. 24
lambda.epsilon ...................................................... 25
lambda_N ............................................................ 26
likelihood.test ..................................................... 26
maxlik.betasplit ................................................... 28
mcmc_alpha ........................................................ 30
mcmc_eta ........................................................... 31
nodes_depths_ordonnes .......................................... 33
plot.treeshape ..................................................... 34
primates ............................................................. 35
rbactrian ........................................................... 35
rhodopsin .......................................................... 35
rtreeshape .......................................................... 36
sackin ............................................................... 38
shape.statistic ..................................................... 39
shift.test .......................................................... 41
simulate.R ......................................................... 42
simulate.R.K ....................................................... 43
simulate.Tau.X ..................................................... 43
simulate.Yi ........................................................ 44
simulate_kingman ................................................ 45
simulate_tree ..................................................... 46
simulate_yule ...................................................... 47
smaller.clade.spectrum .......................................... 48
spectrum.treeshape .............................................. 49
split ................................................................. 50
subtree.test ........................................................ 51
summary.treeshape ............................................... 52
tipsubtree .......................................................... 53
transform ........................................................... 54
transform_eta ....................................................... 55
Description

A graphical test to decide if tree data fit the Yule or the PDA models.

Usage

aldous.test(tree, xmin=20, ...)

Arguments

tree       An object of class "treeshape".

xmin       An object of class "numeric" that defines the range of the x-axis. The minimal
            parent clade size displayed in the graphical representation (default: xmin=20).

...       further arguments to be passed to plot().

Details

A binary tree contains a set of splits

\[(m, i) = (\text{size of parent clade, size of smaller daughter clade})\]

which can be plotted as a scatter diagram. Aldous’ proposal for studying tree balance is that, given
a large phylogenetic tree, one should estimate the median size of the smaller daughter clade as a
function of the parent clade and use this function as a descriptor of balance or imbalance of the tree.
It is convenient to make a log-log plot and to ignore small parent clades. The scatter diagram shows
lines giving the approximate median values of the size of smaller daughter clade predicted by the
beta-splitting model for two values of beta, the value for the Yule \((\beta = 0)\) and PDA \((\beta = -1.5)\)
models. In other words, if the null model were true, then the scatter diagram for a typical tree would
have about half the points above the line and half below the line, throughout the range.

The green line represents the median regression estimated from the tree data.

Value

The function provides a graphical display of results.
all.equal.treeshape

**Description**

This function makes a global comparison of two phylogenetic trees.

**Usage**

```r
## S3 method for class 'treeshape'
all.equal(target, current, names=FALSE, height=FALSE, ...)
```

**Arguments**

- `target` An object of class "treeshape".
- `current` An object of class "treeshape".
- `names` An object of class "logical", checking if the names of the tips should be tested. If FALSE (default), the names of the tips are not compared.
- `height` An object of class "logical", checking if the heights of the nodes should be tested. If FALSE (default), the height of internal nodes are not compared.
- `...` further arguments passed to or from other methods.

**Examples**

```r
library(quantreg)
aldous.test(rbiased(2000, p=.5))

## Test with a huge balanced tree:
aldous.test(rbiased(2000, p=.5))
```
Details

This function is meant to be an adaptation of the generic function \texttt{all.equal} for the comparison of phylogenetic trees. A phylogenetic tree can have many different representations. Permutations between the left and the right daughter clade of a node do not change the corresponding phylogeny, and \texttt{all.equal.treeshape} returns \texttt{TRUE} on two permutated trees.

Value

Returns the logical \texttt{TRUE} if the tree objects are similar up to a permutation of their tips. Otherwise, it returns \texttt{FALSE}. Heights and labels can be taken into account.

Author(s)

Michael Blum \texttt{<michael.blum@imag.fr>}
Nicolas Bortolussi \texttt{<nicolas.bortolussi@imag.fr>}
Eric Durand \texttt{<eric.durand@imag.fr>}
Olivier Francois \texttt{<olivier.francois@imag.fr>}

See Also

\texttt{all.equal} for the generic R function.

Examples

```r
## Trees with permutations
data(carnivora.treeshape)
tree=carnivora.treeshape
tree$merge[8,]=c(tree$merge[8,2],tree$merge[8,1])
all.equal(tree, carnivora.treeshape)

## Trees with different heights
merge=matrix(NA, 3, 2)
merge[,1]=c(-3,-1,2); merge[,2]=c(-4,-2,1);tree1=treeshape(merge)
merge[,1]=c(-1,-3,1); merge[,2]=c(-2,-4,2);tree2=treeshape(merge)
plot(tree1, tree2)
all.equal(tree1, tree2)
all.equal(tree1, tree2, height=TRUE)

## Trees with different names
tree3=treeshape(tree1$merge, c("a", "b", "c", "d"))
tree4=treeshape(tree1$merge, c("1", "2", "3", "4"))
plot(tree3, tree4)
all.equal(tree3, tree4)
all.equal(tree3, tree4, names=TRUE)
```
as.phylo.treeshape  Conversion among tree objects

Description

as.phylo is a generic function - described in the APE package - which converts an object into a tree of class "phylo". This method is an adapation of this generic method to convert objects of class "treeshape" in objects of class "phylo".

Usage

## S3 method for class 'treeshape'
as.phylo(x, ...)

Arguments

x An object of class "treeshape".

... further arguments to be passed to or from other methods.

Value

An object of class "phylo".

Author(s)

Michael Blum <michael.blum@imag.fr>
Nicolas Bortolussi <nicolas.bortolussi@imag.fr>
Eric Durand <eric.durand@imag.fr>
Olivier Francois <olivier.francois@imag.fr>

See Also

as.phylo
as.treeshape

Examples

data(primates)
plot(primates)

library(ape)

primates.phylo = as.phylo(primates)
plot(primates.phylo)
Description

`as.treeshape` is a generic function which converts an tree object into an object of class "treeshape". There is currently a single method for this generic function.

Usage

```r
## S3 method for class 'phylo'
as.treeshape(x, model, p, ...)
```

Arguments

- `x`: An object to be converted into another class. Currently, it must be an object of class "phylo".
- `model`: The model to use when the tree to convert is not binary. If `NULL` (default), the tree is not converted. One of "biased", "pda", "aldous" or "yule" character string.
- `p`: The parameter for the model "biased".
- `...`: Further arguments to be passed to or from other methods.

Details

`as.treeshape` can convert trees that are not binary. When trying to convert a tree with polytomies, this function may either reject the tree (if `model=NULL`) or simulate the tree. The polytomy is replaced by a randomized subtree with n tips where n is the size of the polytomy. The subtree is simulated using the PDA, Yule, Aldous or biased model.

Value

An object of class "treeshape" or an object of classes "treeshape" and "randomized.treeshape" if the original tree was not binary.

Author(s)

Michael Blum <<michael.blum@imag.fr>>
Nicolas Bortolussi <<nicolas.bortolussi@imag.fr>>
Eric Durand <<eric.durand@imag.fr>>
Olivier Francois <<olivier.francois@imag.fr>>

See Also

`as.phylo.treeshape`
Examples

library(ape)
data(bird.orders)
## Data set from APE
plot(bird.orders)

## "treeshape" conversion
tree=as.treeshape(bird.orders)
plot(tree)
summary(tree)

---

aux_lik

*Probability of the sampled node position*

---

**Description**

Probability of the sampled node position knowing the tree topology, the number of unsampled nodes and the interval sizes

**Usage**

aux_lik(M, beta, alpha)

**Arguments**

- **M**: An object returned by the function transform
- **beta**: Imbalance index
- **alpha**: Clade age-richness index

**Author(s)**

Odile Maliet, Fanny Gascuel & Amaury Lambert

---

**a_N**

*Computes a_n*

---

**Description**

Values of a_n for n in 1:N

**Usage**

a_N(beta, N)
**bind.trees**

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>beta</td>
<td>Imbalance index</td>
</tr>
<tr>
<td>N</td>
<td>Tip number</td>
</tr>
</tbody>
</table>

**Details**

\[ a_n = \int_0^1 (1 - r^n - (1 - r)^n)r^\beta(1 - r)^\beta \, dr \]

**Value**

A vector of \( a_n \) for \( n \) in \( 1:N \)

**Author(s)**

Odile Maliet, Fanny Gascuel & Amaury Lambert

**References**


---

**bind.trees**

* Binds two tree together

**Description**

Binds two tree together

**Usage**

`bind.trees(a1, a2, root1, root2, ab = FALSE)`

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>A phylo object</td>
</tr>
<tr>
<td>a2</td>
<td>A phylo object</td>
</tr>
<tr>
<td>root1</td>
<td>The length of the branch between the crown of the resulting tree and the crown of a1</td>
</tr>
<tr>
<td>root2</td>
<td>The length of the branch between the crown of the resulting tree and the crown of a2</td>
</tr>
<tr>
<td>ab</td>
<td>Boolean indicating whether phylogenies come with tip abundance data (default to FALSE)</td>
</tr>
</tbody>
</table>

**Author(s)**

Odile Maliet, Fanny Gascuel & Amaury Lambert
build_tree  

**Description**

Builds the tree using node depths as in the birth-death model and coalescent point process; "The shape and probability of reconstructed phylogenies", by Amaury Lambert and Tanja Stadler (Theoretical Population Biology, 2013)

**Usage**

```r
build_tree(H, tip.lab = rep(NA, length(H) + 1))
```

**Arguments**

- `H` Vector of node depths
- `tip.lab` Vector of tip labels

**Value**

A phylo object

**Author(s)**

Odile Maliet, Fanny Gascuel & Amaury Lambert

**References**


---

carnivora.treeshape  

**Description**

An object of class "treeshape" containing the phylogeny of 12 biological families in the Carnivora.

**Usage**

```r
data(carnivora.treeshape)
```
**change_int**

*Internal function*

**Description**

An auxiliary function used in mcmc_alpha

**Usage**

```r
change_int(enhanced, ind, tree, alpha, beta, epsilon, lambdaN, aN)
```

**Arguments**

- enhanced
- ind
- tree
- alpha
- beta
- epsilon
- lambdaN
- aN

**Author(s)**

Odile Maliet, Fanny Gascuel & Amaury Lambert

---

**change_int_eta**

*Internal function*

**Description**

An auxiliary function used in mcmc_eta

**Usage**

```r
change_int_eta(enhance, ind, tree, alpha, beta, eta, epsilon, lambdaN, aN, change = FALSE, uns)
```
Arguments

- enhance
- ind
- tree
- alpha
- beta
- eta
- epsilon
- lambdaN
- aN
- change
- uns

Author(s)
Odile Maliet, Fanny Gascuel & Amaury Lambert

| cladesize | Compute the number of children of a randomly chosen node |

Description
cladesize takes a random internal node in a tree, and computes its number of descendants (clade size).

Usage
cladesize(tree)

Arguments
- tree An object of class "treeshape".

Details
This function can be used to check whether a tree fits the Yule or the PDA models. Under the Yule model, the probability distribution of the random clade size is equal to

\[ P(K_n = k) = \frac{2n}{(n-1)k(k+1)} \]

for \( k = 2, 3, \ldots, n-1 \) and

\[ P(K_n = n) = \frac{1}{n-1} \]
(where \( n \) is the number of tips of the tree and \( Kn \) is the number of descendents of an internal node of the tree). Under the PDA model, the asymptotic distribution (when the number of tips grows to infinity) of the random clade size is equal to

\[
P(K = k + 1) = \frac{\binom{2k}{k}}{(k + 1)(2^k)^2}
\]

Value

An object of class numeric representing the clade size of a random node of a tree.

Author(s)

Michael Blum <michael.blum@imag.fr>
Nicolas Bortolussi <nicolas.bortolussi@imag.fr>
Eric Durand <eric.durand@imag.fr>
Oliver Francois <olivier.francois@imag.fr>

References

Blum, M., Francois, O. and Janson, S. The mean, variance and limiting distribution of two statistics sensitive to phylogenetic tree balance; manuscript available from http://www-timc.imag.fr/Olivier.Francois/bfj.pdf.

Examples

# Histogram of random clade sizes
main="Random clade sizes for random generated trees"
xlabel="clade size"
hist(sapply(rtreeshape(100, tip.number=40, model="yule"), FUN=cladesize),
freq=FALSE, main=main, xlab=xlabel)

colless <- Compute the Colless’ shape statistic on tree data

Description

colless computes the Colless’ index of a tree and provides standardized values according to the Yule and PDA models.

Usage

colless(tree, norm = NULL)
Arguments

- **tree**: An object of class "treeshape" on which the Colless’ index is computed.
- **norm**: A character string equals to NULL (default) for no normalization or one of "pda" or "yule".

Details

The Colless’ index $I_c$ computes the sum of absolute values $|L - R|$ at each node of the tree where $L$ (resp. $R$) is the size of the left (resp. right) daughter clade at the node.

The mean and standard deviation of the Colless’s statistic $I_c$ have been computed by Blum et al (2005). Under the Yule model the standardized index

$$I_{yule} = \frac{I_c - n \star \log(n) - n(\gamma - 1 - \log(2))}{n}$$

converges in distribution ($\gamma$ is the Euler constant). The limiting distribution is non Gaussian and is characterized as a functional fixed-point equation solution. Under the PDA model, the standardization is different

$$I_{\text{pda}} = \frac{I_c}{n^{3/2}}$$

and converges in distribution to the Airy distribution (See Flajolet and Louchard (2001)). Standardized indices are useful when one wishes to compare trees with different sizes. The colless function returns the value of the unnormalized index (default) or one of the standardized statistics (Yule or PDA).

Value

An object of class numeric which is the Colless’ index of the tree.

Author(s)

Michael Blum <michael.blum@imag.fr>
Nicolas Bortolussi <nicolas.bortolussi@imag.fr>
Eric Durand <eric.durand@imag.fr>
Olivier Francois <olivier.francois@imag.fr>

References


See Also

sackin
**cutreeshape**

**Examples**

```r
## Colless' index for a randomly generated PDA tree (unnormalized value)
tpda<-rtreeshape(1,tip.number=70,model="pda")
colless(tpda[[1]],norm="pda")

## Histogram of Colless' indices for randomly generated Yule trees
main="Colless' indices for randomly generated Yule trees"

xlab="Colless' indices"

hist(sapply(rtreeshape(50,tip.number=50,model="yule"),FUN=colless,norm="yule"),
 freq=FALSE,main=main,xlab=xlab)

## Change the number of tips
hist(sapply(rtreeshape(50,tip.number=25,model="yule"),FUN=colless,norm="yule"),
 freq=FALSE,main=main,xlab=xlab)
```

**cutreeshape**  
*Cut objects of class "treeshape"*

**Description**

Prunes or cuts an object of class "treeshape" from a specified internal node, either by specifying a top or bottom direction. This function returns either the top part or the bottom part of a tree.

**Usage**

`cutreeshape(tree, node, type)`

**Arguments**

- `tree`: An object of class "treeshape".
- `node`: An integer representing the node at which the tree will be pruned or cut. `node` should be in the range 1, ..., `treesize` - 1.
- `type`: A character string equals to either "top" or "bottom".

**Details**

If the `type` argument is "top", the tree is pruned from `node`. The resulting tips correspond to the ancestral branches present at the same height as the given node. New tip labels are assigned to the tips.

If the `type` specified is "bottom", the subtree under `node` is returned. The tips are not renamed (they keep their former names) and the specified node is the root of the new tree.

**Value**

An object of class "treeshape"
Author(s)

Michael Blum <michael.blum@imag.fr>
Nicolas Bortolussi <nicolas.bortolussi@imag.fr>
Eric Durand <eric.durand@imag.fr>
Olivier Francois <olivier.francois@imag.fr>

See Also

tipsubtree

Examples

```r
## Data set provided with the library. Type help(cytochromc) for more infos.
data(carnivora.treeshape)
data(hivtree.treeshape)

## Examples of "bottom" cutting:
bottom.tree=cutreeshape(carnivora.treeshape, 3, "bottom")
plot(carnivora.treeshape, bottom.tree)
bottom.tree=cutreeshape(carnivora.treeshape, 8, "bottom")
plot(carnivora.treeshape, bottom.tree)

## Examples of "top" pruning:
top.tree=cutreeshape(hivtree.treeshape, 158, "top")
plot(hivtree.treeshape, top.tree)
```

cythromc  Phylogeny of the cytochrome C family.

description

An object of class "treeshape" containing the phylogeny of 50 species of cytochrome c.

depth  Gets the node depths of the tree

description

Gives the node depths of the tree, in the order of node labels

Usage

depth(tree)
**enhance**

**Arguments**

- `tree` A phylo object

**Author(s)**

Odile Maliet, Fanny Gascuel & Amaury Lambert

---

**Description**

Enhance the data by drawing intervals at nodes

**Usage**

```
enhance(tree, alpha, beta, epsilon, lambdaN, aN)
```

**Arguments**

- `tree`
- `alpha`
- `beta`
- `epsilon`
- `lambdaN`
- `aN`

**Author(s)**

Odile Maliet, Fanny Gascuel & Amaury Lambert

---

**enhance_eta**

**Internal function**

---

**Description**

Enhance the data by drawing intervals at nodes

**Usage**

```
enhance_eta(tree, alpha, beta, eta, epsilon, lambdaN, aN, uns)
```
get_extinction_list

**Arguments**

- tree
- alpha
- beta
- eta
- epsilon
- lambdaN
- aN
- uns

**Author(s)**

Odile Maliet, Fanny Gascuel & Amaury Lambert

---

**get_extinction_list**  
*Gives the tips of the phylogeny in their extinction order*

---

**Description**

Orders the tips which will sequentially get extinct/be sampled, based on relative species abundance as explained in the main text of the article.

**Usage**

```r
get_extinction_list(N, tree, equal.ab = TRUE)
```

**Arguments**

- N  
  Tip nuber
- tree  
  A phylo object
- equal.ab  
  If set to TRUE (the default), all species have the same probability to go extinct first

**Author(s)**

Odile Maliet, Fanny Gascuel & Amaury Lambert

**References**

get_PD_sample

Computes the proportion of conserved PD

Description

Computes the proportion of conserved phylogenetic diversity as a function of the proportion of conserved species, in trees simulated by the model.

Usage

```
get_PD_sample(epsilon, beta, alpha, N, sampl.frac, ntree, equal.ab, eta, lengths = "yule", b = 1, d = 0)
```

Arguments

- `epsilon`: Minimum size of unsampled splits (see appendix 1)
- `beta`: Imbalance index
- `alpha`: Clade age-richness index
- `N`: Initial tip number
- `sampl.frac`: Vector of tips fractions for which we want to compute the conserved PD
- `ntree`: Number of simulated trees
- `equal.ab`: If set to TRUE, all species have the same probability to go extinct first (default to TRUE)
- `eta`: Clade abundance-richness index (if equal.ab == FALSE)
- `lengths`: Model used to simulate node depths (can be "yule" (the default) or "kingman")
- `b`: Birth rate (if lengths == "yule")
- `d`: Death rate (if lengths == "yule")

Value

A table of size length(sample.fract)*ntree. The element in ligne i and column j is the remaining Phylogenetic Diversity fraction for the j*th tree in which a fraction sample.frac[i] has been sampled.

Author(s)

Odile Maliet, Fanny Gascuel & Amaury Lambert

References

Examples

set.seed(813)
sampl.frac=seq(0,1,by=0.05)

PD = get_PD_sample(epsilon=0.01,beta=0,alpha=0,N=50,
    sampl.frac=sampl.frac,ntree=10,equal.ab=FALSE,
    eta=0.5,lengths="yule",b=1,d=0)
probs = c(0.1, 0.9, 0.5)
PD_stats = as.vector(t(sapply(1:nrow(PD), function(i){quantile(PD[i,], probs)})))
par(mgp=c(2.2, 0.8, 0))
par(mar=c(4, 4, 1, 1))
plot(1, type="n", xlab="Fraction of extinct species, p",
    ylab="PD loss", ylim=c(0,1), xlim=c(0,1))
points(c(0,1),c(0,1),t="l",col=grey(0),lty=3)
# plot 95% confidence intervals (grey area)
polygon(c(1-sampl.frac, rev(1-sampl.frac)), c(1-PD_stats[1:length(sampl.frac)]),
    rev(1-PD_stats[((length(sampl.frac)+1):(2*length(sampl.frac)))]),
    border=NA, col=grey(0.7))
# plot median value (black line)
points(1-sampl.frac, 1-PD_stats[((2*length(sampl.frac)+1):(3*length(sampl.frac)))]),t="l")

---

get_tree_beta

Beta parameter as a function of the proportion of remaining tips

Description

Computes the maximum likelihood estimate of the parameter beta in trees simulated by the model,
as a function of the proportion of conserved species

Usage

get_tree_beta(epsilon, beta, alpha, N, sampl.frac, ntree, equal.ab = TRUE, eta = 1)

Arguments

epsilon Minimum size of unsampled splits (see appendix 1)

beta Imbalance index

alpha Clade age-richness index

N Initial tip number

sampl.frac Vector of tips fractions for which we want to compute the beta statistic

ntree Number of simulated trees

equal.ab If set to TRUE, all species have the same probability to go extinct first (default
to TRUE)

eta Clade abundance-richness index (if equal.ab == FALSE)
Value

A table of size length(sample.fract)*ntree. The element in ligne i and column j is the Maximum Likelihood Estimate for the beta statistic of the j^th tree in which a fraction sample.frac[i] has been sampled.

Author(s)

Odile Maliet, Fanny Gascuel & Amaury Lambert

References


Examples

# With the field of bullets hypothesis

set.seed(813)
sampl.frac=seq(0.2,1,0.2)
Beta=get_tree_beta(epsilon=0.01, beta=-1, alpha=-1, N=20, sampl.frac=sampl.frac, ntree=3)

Beta_quantiles=sapply(1:nrow(Beta),function(x){quantile(Beta[x,],c(0.05,0.5,0.95)))

plot(1, type="n", xlab="Fraction of extinct species, p", ylab="Beta statistic", ylim=c(-2,10), xlim=c(0,1))
polygon(c(1-sampl.frac, rev(1-sampl.frac)), c(Beta_quantiles[1,(1:length(sampl.frac))], rev(Beta_quantiles[3,(1:length(sampl.frac))])), border=NA, col=grey(0.7))
points(1-sampl.frac, Beta_quantiles[2,(1:length(sampl.frac))],t="l")

# With nonrandom extinctions

## Not run:
set.seed(813)
sampl.frac=seq(0.2,1,0.2)
Beta=get_tree_beta(epsilon=0.01, beta=5, alpha=2, eta=2, N=20, sampl.frac=sampl.frac, ntree=3)

Beta_quantiles=sapply(1:nrow(Beta),function(x){quantile(Beta[x,],c(0.05,0.5,0.95)))

plot(1, type="n", xlab="Fraction of extinct species, p", ylab="Beta statistic", ylim=c(-2,10), xlim=c(0,1))
polygon(c(1-sampl.frac, rev(1-sampl.frac)), c(Beta_quantiles[1,(1:length(sampl.frac))], rev(Beta_quantiles[3,(1:length(sampl.frac))])), border=NA, col=grey(0.7))
points(1-sampl.frac, Beta_quantiles[2,(1:length(sampl.frac))],t="l")

## End(Not run)
hivtree.treeshape  
Phylogenetic Tree of 193 HIV-1 Sequences

Description
This data set describes an estimated clock-like phylogeny of 193 HIV-1 group M sequences sampled in the Democratic Republic of Congo. This data is the conversion of the data from the APE package into the class "treeshape".

Usage
data(hivtree.treeshape)

Format
hivtree.treeshape is an object of class "treeshape".

Source
This is a data example from Strimmer and Pybus (2001).

References

Examples

data("hivtree.treeshape")
summary(hivtree.treeshape)
plot(hivtree.treeshape)

index.test

Perform a test on the Yule or PDA hypothesis based on the Colless or the Sackin statistic

Description
colless.test performs a test based on the Colless’ index on tree data for the Yule or PDA model hypothesis.
sackin.test does the same with the Sackin’s index.
Usage

colless.test(tree, model = "yule", alternative = "less", n.mc = 500)
sackin.test(tree, model = "yule", alternative = "less", n.mc = 500)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tree</td>
<td>An object of class &quot;treeshape&quot;.</td>
</tr>
<tr>
<td>model</td>
<td>The null hypothesis of the test. One of &quot;yule&quot; (default) or &quot;pda&quot;.</td>
</tr>
<tr>
<td>alternative</td>
<td>A character string specifying the alternative hypothesis of the test. One of &quot;less&quot; (default) or &quot;greater&quot;.</td>
</tr>
<tr>
<td>n.mc</td>
<td>An integer representing the number of random trees to be generated and required to compute a p-value from a Monte Carlo method.</td>
</tr>
</tbody>
</table>

Details

A test on tree data that either rejects the Yule or the PDA models. This test is based on a Monte Carlo estimate of the p-value. Replicates are generated under the Yule or PDA models, and their Colless’ (Sackin’s) indices are computed. The empirical distribution function of these statistics is then computed thanks to the "ecdf" R function. The p-value is then deduced from its quantiles. The less balanced the tree is and the larger its Colless’s (Sackin’s) index. The alternative "less" should be used to test whether the tree is more balanced (less unbalanced) than predicted by the null model. The alternative "greater" should be used to test whether the tree is more unbalanced than predicted by the null model. The computation of p-values may take some time depending on the number of replicates n.mc and the size of the simulated tree.

Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>model</td>
<td>the null model.</td>
</tr>
<tr>
<td>statistic</td>
<td>the test statistic.</td>
</tr>
<tr>
<td>p.value</td>
<td>the p-value of the test.</td>
</tr>
<tr>
<td>alternative</td>
<td>the alternative hypothesis of the test.</td>
</tr>
</tbody>
</table>

Author(s)

Michael Blum <<michael.blum@imag.fr>>
Nicolas Bortolussi <<nicolas.bortolussi@imag.fr>>
Eric Durand <<eric.durand@imag.fr>>
Olivier Francois <<olivier.francois@imag.fr>>

References


See Also

subtree.test
colless
sackin

Examples

```r
## Test on a randomly generated Yule tree with 30 tips
a<-rtreeshape(1,30,model="yule")
a<-a[[1]]

## Is it more balanced than a Yule tree ?
colless.test(a,alternative="less",model="yule")
## Is it less balanced than a PDA tree ?
colless.test(a,model="pda",alternative="greater")

## Test on the phylogenetic tree hiv.treeshape: is it more balanced than
## predicted by the Yule model?
data(hivtree.treeshape)
## The tree looks compatible with the null hypothesis
colless.test(hivtree.treeshape, alternative="greater", model="yule")

## What happen when we look at the top the tree?
colless.test(cutreeshape(hivtree.treeshape, 160, "top"),
    alternative="greater", model="yule")
colless.test(cutreeshape(hivtree.treeshape, 160, "top"),
    alternative="greater", model="pda")

## Test with the Sackin's index: is the HIV tree less balanced than
## predicted by the PDA model?
sackin.test(hivtree.treeshape,alternative="greater",model="pda")
## The p.value equals to 1...
```

---

**insert**

*Insert an element in a vector*

**Description**

Insert an element in a vector

**Usage**

```r
insert(v, e, i, replace)
```
**lambda.epsilon**

**Arguments**

- **v**: Vector
- **e**: Element to be inserted
- **i**: Position where the element should be inserted
- **replace**: Should the element at position i be replaced by e?

**Author(s)**

Odile Maliet, Fanny Gascuel & Amaury Lambert

---

**lambda.epsilon**  
*Computes lambda.epsilon*

---

**Description**

Internal function needed to simulate the ranked tree shape model

**Usage**

lambda.epsilon(epsilon, beta, n, upper = 10000)

**Arguments**

- **epsilon**: Minimum size of unsampled splits (see appendix 1)
- **beta**: Imbalance index
- **n**: Tip number
- **upper**: Upper bound from the integral (see lambda_n expression, default to 10000)

**Details**

\[ \lambda_\epsilon = 2 \int_{\epsilon}^{\infty} e^{-x(\beta + n + 1)}(1 - e^{-x})^\beta \, dx \]

**Author(s)**

Odile Maliet, Fanny Gascuel & Amaury Lambert

**References**

**lambda_N**

**Computation of lambda_epsilon**

**Description**

Computes the parameter lambda_epsilon for n in 1:N

**Usage**

`lambda_N(epsilon, beta, N, upper = 10000)`

**Arguments**

- `epsilon` Minimum size of unsampled splits (see appendix 1)
- `beta` The beta parameter quantifying the balance of the tree
- `N` Tip number
- `upper` Upper bound from the integral (see lambda_n expression, default to 10000)

**Details**

\[
\lambda_\epsilon = 2 \int_\epsilon^\infty \exp(-\beta n - 1) x (1 - \exp(-x))^3 dx
\]

**Author(s)**

Odile Maliet, Fanny Gascuel & Amaury Lambert

**References**


---

**likelihood.test**

**Test the Yule model vs PDA (uniform) model.**

**Description**

`likelihood.test` uses the function `shape.statistic` to test the Yule model against the PDA model. The test is based on a Gaussian approximation for the log-ratio of likelihoods.

**Usage**

`likelihood.test(tree, model = "yule", alternative="two.sided")`
Arguments

- **tree**: An object of class "treeshape" on which the test is performed.
- **model**: The null hypothesis of the test. It must be equal to one of the two character strings "yule" or "pda".
- **alternative**: A character string specifying the alternative hypothesis of the test. Must be one of "two.sided" (default), "less" or "greater".

Details

A test on tree data that either rejects the Yule of the PDA model. The test is based on the ratio of the likelihood of the PDA model to the likelihood of the Yule model (shape.statistic). The less balanced the tree is the larger its shape statistic is. The alternative "less" should be used to test whether the tree is less unbalanced than predicted by the null model. The alternative "greater" should be used to test whether the tree is more unbalanced than predicted by the null model.

Under the Yule model, the test statistic has approximate Gaussian distribution of mean $1.204 \times n - \log n - 1 - 2$ and variance $0.168 \times n - 0.710$, where $n$ is the number of tips of the tree. The Gaussian approximation is accurate for $n$ greater than 20.

Under the PDA model, the test statistic has approximate Gaussian distribution of mean $2.03 \times n - 3.545 \times \sqrt{n} - 1$ and variance $2.45 \times (n - 1) \times \log n - 1$, where $n$ is the number of tips of the tree. The Gaussian approximation is however accurate for very large $n$ ($n$ greater than 10000(?)). The values of the means and variances have been obtained from an analogy with binary search tree models in computer science.

The function includes corrections for small sizes under the PDA model, and uses empirical values of variances estimated through Monte Carlo replicates as follows

$$variance \sim 1.570 \times n \times \log n - 5.674 \times n + 3.602 \times \sqrt{n} + 14.915$$

Value

`likelihood.test` returns a list which includes:

- **model**: the null model used by the test
- **statistic**: the test statistic
- **p.value**: the p.value of the test
- **alternative**: the alternative hypothesis of the test

Author(s)

Michael Blum <<michael.blum@imag.fr>>
Nicolas Bortolussi <<nicolas.bortolussi@imag.fr>>
Eric Durand <<eric.durand@imag.fr>>
Olivier Francois <<olivier.francois@imag.fr>>
maxlik.betasplit

Maximum likelihood of beta in the Beta-splitting model

Description

The function finds the beta value that maximizes the likelihood in the Beta-splitting model. Beta=0 corresponds to Yule trees, Beta<0 corresponds to trees more unbalanced than Yule trees and Beta>0 corresponds to trees more balanced than Yule trees. Confidence intervals can also be provided.

Usage

maxlik.betasplit(phylo, up = 10, remove.outgroup = FALSE, confidence.interval = "none", conf.level = 0.95, size.bootstrap = 100)

Arguments

phylo An object of class "treeshape" or "phylo" on which the likelihood is computed.
up numeric. The Beta value that maximizes the likelihood is searched between -2 and up
remove.outgroup logical. Should one or two outgroups be removed before computing the likelihood
confidence.interval The method to be used for computing confidence intervals. See *Details*.
conf.level numeric. Probability associated with the confidence interval
size.bootstrap number. Number of bootstrap replicates

References


See Also

shape.statistic

Examples

```r
## Generate a Yule tree with 150 tips. Is it likely to be fitting the PDA model?
likelihood.test(ryule(150),model="pda")
## The p.value is close from 0. We reject the PDA hypothesis.

## Test on the Carnivora tree: is it likely to be fitting the Yule model?
data(carnivora.treeshape)
likelihood.test(carnivora.treeshape)
## The p.value is high, so it's impossible to reject the Yule hypothesis.
```
Details

The beta-splitting model has been introduced by Aldous to simulate trees with different tree balance. Beta=0 corresponds to Yule trees.
Beta<0 corresponds to trees more unbalanced than Yule trees, Beta=-3/2 corresponds to the PDA model.
Beta>0 corresponds to trees more balanced than Yule trees.
By default, confidence.interval="none" and no confidence interval is computed.
When confidence.interval="bootstrap", a confidence interval is found with a resampling technique. Shall be used when the number of tips is small (<50).
When confidence.interval="profile", a confidence interval is found with a profile likelihood technique. Shall be used when the number of tips is large (>50).

Value

max_lik The Beta value that maximizes the likelihood
conf_interval A confidence interval for max_lik

Author(s)

michael.blum@imag.fr

References


See Also

sackin, sackin.test, colless, colless.test

Examples

tree.pda<-rtreeshape(n=1, tip.number=50, model="pda")[[1]]
maxlik.betasplit(tree.pda,confidence.interval="none")
#bootstrap example is commented because it is too slow to run
#maxlik.betasplit(tree.pda,confidence.interval="bootstrap")
maxlik.betasplit(tree.pda,confidence.interval="profile")
Inference of the alpha parameter

Description

Run the Bayesian inference of the clade age-richness index alpha

Usage

mcmc_alpha(tree, epsilon, beta, niter, ini = 0, V = 0.1,
            chain = NULL, verbose = 10, silent = TRUE, Nadapt = Inf,
            NadaptMin = 10, NadaptMax=Inf, ma = -4, Ma = 4, proposal = "bactrian",
            accOpt = 0.3, Vmin = 0.001)

Arguments

- tree: A phylo object
- epsilon: Minimum size of unsampled splits (see appendix 1)
- beta: Imbalance index
- niter: Number of iterations in the mcmc
- ini: Initial alpha value (default to 0)
- V: Initial scaling value for the mcmc proposal (default to 0.1)
- chain: Former mcmc chain (if NULL (the default), a new one is started)
- verbose: Number of iterations after which the state of the mcmc is printed (if silent == FALSE)
- silent: If TRUE (the default) the state of the mcmc is not printed
- Nadapt: Number of iterations between each proposal scalling (default to Inf)
- NadaptMin: Minimum number of iterations before the first proposal scalling (default to 10)
- NadaptMax: Number of iterations after which the proposal stops being scalled (default to Inf)
- ma: Minimal alpha value (default to -4)
- Ma: Maximal alpha value (default to 4)
- proposal: Shape of the proposal. Can be "bactrian" (the default, ref), "uniform", or "normal"
- accOpt: Optimal acceptance value (default to 0.3)
- Vmin: Minimal scaling value for the mcmc proposal (default to 0.001)

Value

A list with a mcmc field containing the resulting chain. The other fields are only used to resume running the inference if the chain has to be completed.
mcmc_eta

Author(s)
Odile Maliet, Fanny Gascuel & Amaury Lambert

References

Examples
```r
ntip=30
set.seed(123)
tree=simulate_tree(epsilon = 0.01, alpha = -1, beta = 0, N = ntip, equal.ab = TRUE)
beta=maxlik.betasplit(tree,up=10)$max_lik
plot(tree)

niter=1000

## Not run:
chain=mcmc_alpha(tree, epsilon=0.01, beta=beta, niter=600, V = c(0.1), ini=c(0),
veroese = 100, silent = FALSE, Nadapt = 100, NadaptMin = 100)

# Continue the same chain
chain=mcmc_alpha(tree, epsilon=0.01, beta=beta, niter=400, verbose = 100, silent = FALSE,
chain = chain, Nadapt = 100, NadaptMin = 500, NadaptMax = 700)

thinned=mcmc(chain$mcmc[seq(200,1000,10),])
plot(thinned)
da=density(thinned[,"alpha"])
MPa=da$x[which.max(da$y)]
print(MPa)

## End(Not run)
```

mcmc_eta

Inference of the alpha and eta parameters

Description
Run the Bayesian inference of the clade age-richness index alpha and the clade abundance richness index eta

Usage
```r
mcmc_eta(tree, epsilon, beta, ini = c(0, 1), V = c(0.1, 0.1), chain = NULL, niter,
verbose = 10, silent = TRUE, Nadapt = Inf, NadaptMin = 10, NadaptMax=Inf,
ma = -4, Ma = 4, me = 0.1, Me = 10, proposal = "bactrian", accOpt = 0.3)
```
Arguments

- `tree`: A phylo object
- `epsilon`: Minimum size of unsampled splits (see appendix 1)
- `beta`: Imbalance index
- `ini`: Initial alpha and eta values (default to c(0,1))
- `V`: Initial scaling value for the mcmc proposal (default to c(0.1,0.1))
- `chain`: Former mcmc chain (if NULL (the default), a new one is started)
- `niter`: Number of iterations in the mcmc
- `verbose`: Number of iterations after which the state of the mcmc is printed (if silent == FALSE)
- `silent`: If TRUE (the default) the state of the mcmc is not printed
- `Nadapt`: Number of iterations between each proposal scaling (default to Inf)
- `NadaptMin`: Minimum number of iterations before the first proposal scaling (default to 10)
- `NadaptMax`: Number of iterations after which the proposal stops being scaled (default to Inf)
- `ma`: Minimal alpha value (default to -4)
- `Ma`: Maximal alpha value (default to 4)
- `me`: Minimal eta value (default to 0.1)
- `Me`: Maximal eta value (default to 10)
- `proposal`: Shape of the proposal. Can be "bactrian" (the default, ref), "uniform", or "normal"
- `accOpt`: Optimal acceptance value (default to 0.3)

Value

Returns a list with a mcmc field containing the resulting chain. The other fields are only used to resume running the inference if the chain has to be completed.

Author(s)

Odile Maliet, Fanny Gascuel & Amaury Lambert

References


Examples

```r
seed=123
set.seed(seed)
ntip=30
tree=simulate_tree(epsilon = 0.001, alpha = -1, beta = 0, N = ntip, equal.ab = FALSE, eta = 1.5)
beta=maxlik.betasplit(tree, up=10)$max_lik
extinctions = rank(tree$tip.ab)
```
nodes_depthsordonnes

Gets the node depths of the tree

Description

Gives the node depths of the tree, in the order needed to reconstruct the tree using build.tree

Usage

nodes_depthsordonnes(tree)

Arguments

tree A phylo object

Author(s)

Odile Maliet, Fanny Gascuel & Amaury Lambert
**plot.treeshape**  
*Plot phylogenetic treeshapes.*

**Description**

Plot method for objects of class "treeshape".

**Usage**

```r
## S3 method for class 'treeshape'
plot(x, y, ...)
```

**Arguments**

- `x`: An object of class "treeshape".
- `y`: An object of class "treeshape".
- `...`: Further arguments to be passed to `plot()`.

**Details**

If two trees are specified, they are plotted on the same window. This option is provided in order to facilitate the comparison between two trees.

**Value**

A null value is returned. Results are displayed on graphical window.

**Author(s)**

Michael Blum <<michael.blum@imag.fr>>
Nicolas Bortolussi <<nicolas.bortolussi@imag.fr>>
Eric Durand <<eric.durand@imag.fr>>
Olivier Francois <<olivier.francois@imag.fr>>

**See Also**

`plot` for the basic plotting function in R

**Examples**

```r
## Visual representation of the universal tree of life provided in data
data(universal.treeshape)
plot(universal.treeshape)

## Visual representation of two trees at the same time
data(carnivora.treeshape)
plot(carnivora.treeshape, cutreeshape(carnivora.treeshape, 8, "bottom"))
```
primates

Description
An object of class "treeshape" containing the phylogeny of the primates.

rbactrian

Description
Proposal for the mcmc functions

Usage
rbactrian(n, m = 0.95)

Arguments
n
m

Author(s)
Odile Maliet, Fanny Gascuel & Amaury Lambert

References

rhodopsin

Description
An object of class "treeshape" containing the phylogeny of 64 rhodopsin proteins.
rtreeshape

Generate a list of random binary trees according to a given model

Description

This function generates a tree or a list of trees of class "treeshape" according to the Yule, PDA, Biased models. Speciation-specified models are also allowed.

Usage

rtreeshape(n, tip.number, p = 0.3, model="", FUN="")

Arguments

n
The number of trees to generate.
tip.number
The number of tips of the trees to generate. Can be a vector.
p
Only used when model="biased". It represents the bias factor of the tree to generate.
model
A character string equals to "yule", "pda", "aldous" or "biased".
FUN
A two variables (n and i) function.

Details

The "FUN" and "model" arguments cannot be specified at the same time. An error will be returned if both arguments are specified.

If tip.number is a vector, n trees will be generated for each size contained by tip.number
Q enables you to build trees of class "treeshape" according to the Markov branching model described by D. Aldous. Qn(i) is the probability that the left daughter clade of an internal node with n descendents contains i tips. The Qn(i) need not sum to one. Still, be careful when you specify this distribution: computational errors may occur for complicated distributions and/or large trees.

The Yule model, also known as Markov model, can be described as follows. At any time, all the extant branches have the same probability to split into two subspecies.
The PDA model (Proportional to Distinguishable Arrangements) is not a model of growing tree. Instead, each tree with n tips has the same probability to be generated under this model. There is \((2n - 3)!!\) possible trees with n tips.
The Biased model is a model of growing tree. When a species with speciation rate r splits, one of its descendents is given the rate pr and the other is given the speciation rate 1 − pr where p is a probability parameter. The Biased model was introduced by Kirkpatrick and Slatkin (1993). The Aldous' Branching (AB) model is defined by the following symmetric split distribution
\[ q(n, i) = \frac{n}{2 \ast h(n - 1)} \ast \left( \frac{1}{i(n - i)} \right), \]
where h(n) is the nth harmonic number. The AB model is hardly motivated by biological considerations.
Value

A list of objects of class "treeshape" NULL if \( n = 0 \)

Author(s)

Michael Blum <michael.blum@imag.fr>
Nicolas Bortolussi <nicolas.bortolussi@imag.fr>
Eric Durand <eric.durand@imag.fr>
Olivier Francois <olivier.francois@imag.fr>

References


Examples

```r
## Summary of a PDA tree with 10 tips:
summary(rtreeshape(n=1, tip.number=10, model="pda")[[1]])
## Summary of a Yule tree with 10 tips:
summary(rtreeshape(n=1, tip.number=100, model="yule")[[1]])

## Generate trees with different sizes
trees=rtreeshape(n=2, tip.number=c(10,20), model="yule")
length(trees)
plot(trees[[1]])
plot(trees[[2]])

## Histogram of Colless’ indices for a list of 100 PDA trees with 50 tips
hist(sapply(rtreeshape(100,50,model="pda"),FUN=colless,norm="pda"),freq=FALSE)

## Histogram of shape statistics for a list of 100 Yule trees with 50 tips
## (takes some time to compute)
main="Histogram of shape statistics for a list of 100 Yule trees"
hist(sapply(rtreeshape(100,50,model="yule"),FUN=shape.statistic,norm="yule"),
freq=FALSE, main=main)
## It should be a gaussian with mean 0 and standard deviation 1
## consider to increase the number of trees>100
x<-seq(-4,4,by=0.01)
lines(x,dnorm(x))

## Building a tree using Markov splitting model
Q <- function(n,i) (i==1)
```
```r
sackin(tree, norm = NULL)
```

**Arguments**

- `tree`: An object of class `"treeshape"` on which the Sackin’s index is computed.
- `norm`: A character string equals to "null" (default) for no normalization, "pda" for the PDA normalization or "yule" for the Yule normalization.

**Details**

The Sackin’s index is computed as the sum of the number of ancestors for each tips of the tree. The less balanced a tree is and the larger its Sackin’s index. It can be normalized in order to obtain a statistic that does not depend on the tree size, and so compare trees with different sizes. The normalization depends on the reference model (Yule or PDA). Under the Yule model, the normalized index is

$$I_{yule} = \frac{I_s - 2n \cdot \sum_{j=2}^{n} \frac{1}{j}}{n}$$

where $I_s$ is the non-normalized Sackin’s index for a n-tips tree. Under the PDA model, the normalized index is

$$I_{pda} = \frac{I_s}{n^{3/2}}$$

See details on the Colless index.

**Value**

An object of class `numeric` which contains the Sackin’s index of the tree.

**Author(s)**

Michael Blum <<michael.blum@imag.fr>>
Nicolas Bortolussi <<nicolas.bortolussi@imag.fr>>
Eric Durand <<eric.durand@imag.fr>>
Olivier Francois <<olivier.francois@imag.fr>>
shape.statistic

References


Blum, M., Francois, O. and Janson, S. The mean, variance and limiting distribution of two statistics sensitive to phylogenetic tree balance; manuscript available from http://www-timc.imag.fr/Olivier.Francois/bfj.pdf.

See Also
colless
sackin.test

Examples

```r
## Index of Sackin of a PDA tree :
tpda<-rtreeshape(1,tip.number=70,model="pda")
tpda<-tpda[[1]]
sackin(tpda,norm="pda")

## Histogram of the Sackin's indices for randomly generated Yule trees,
## with no normalization
main="Histogram of Sackin's indices for randomly generated Yule trees"
xlab="Sackin's index"
hist(sapply(rtreeshape(300,tip.number=50,model="yule"),FUN=sackin,norm="yule"),
    freq=FALSE, main=main, xlab=xlab)

## Change the size of the trees:
hist(sapply(rtreeshape(300,tip.number=100,model="yule"),FUN=sackin,norm="yule"),
    freq=FALSE, main=main, xlab=xlab)
```

**shape.statistic**

Computes the log of the likelihood ratio (yule/pda)

Description

shape.statistic computes the logarithm of the ratio of the likelihoods under the Yule model and the PDA model of the given tree.

Usage

```r
shape.statistic(tree, norm=NULL)
```

Arguments

- `tree`: An object of class "treeshape".
- `norm`: A character string equals to NULL for no normalization, "yule" for the Yule model model normalization or "pda" for the pda normalization.
Details

The log of the likelihood ratio is proportional to

$$\sum (\log N(v) - 1),$$

for all internal node $v$ (where $N(v)$ is the number of internal nodes descending from the node $v$). The ratio of the likelihoods enables to build the most powerful test of the Yule model against the PDA one. (Neyman-Pearson lemma).

Under the PDA model, the log ratio has approximate Gaussian distribution of mean $\sim 2.03 \times n - 3.545 \times \sqrt{n - 1}$ and variance $\sim 2.44 \times (n - 1) \times \log n - 1$, where $n$ is the number of tips of the tree. The Gaussian approximation is accurate for very large $n$ ($n$ greater than 10000(?)). The normalization of the ratio uses tabulated empirical values of variances estimated from Monte Carlo simulations. The normalization uses the formula:

$$\text{variance} \sim 1.570 \times n \times \log n - 5.674 \times n + 3.602 \times \sqrt{n} + 14.915$$

Under the Yule model, the log ratio has approximate Gaussian distribution of mean $\sim 1.204 \times n - \log n - 1 - 2$ and variance $\sim 0.168 \times n - 0.710$, where $n$ is the number of tips of the tree. The Gaussian approximation is accurate for $n$ greater than 20.

Value

An object of class numeric containing the shape statistic of the tree.

Author(s)

Michael Blum <michael.blum@imag.fr>
Nicolas Bortolussi <nicolas.bortolussi@imag.fr>
Eric Durand <eric.durand@imag.fr>
Olivier François <olivier.francois@imag.fr>

References


Examples

data(universal.treeshape)
tree <- universal.treeshape
plot(tree)
summary(tree)

likelihood.test(tree, model = "yule", alternative = "two.sided")
likelihood.test(tree, model = "pda", alternative = "two.sided")

## Histogram of shape statistics for a list of Yule trees
## (may take some time to compute)
main="Histogram of shape statistics"; xlab="shape statistic"
 hist(sapply(rtreeshape(100,tip.number=50,model="yule"),FUN=shape.statistic,
shift.test

Testing diversification rate variation in phylogenetic trees

Description

A statistical test of diversification rate shift within an evolutionary process.

Usage

shift.test(tree, node, lambda1, lambda2, nrep, silent)

Arguments

tree An object of class treeshape.
node An integer value that represents the internal node at which the test is done. It ranges from 1 to the number of tips minus 2.
lambda1 A positive numerical value (object of class numeric) that represents an ancestral diversification rate in the lineage ending at node.
lambda2 A positive numerical value (object of class numeric) that represents the shifted diversification rate after the speciation event at node.
nrep Number of Monte-carlo replicates in the computation of the P-value.
silent A boolean indicating whether the test is silent or verbose.

Details

This function implements a test of diversification rate shift based on the Delta1 statistic (Moore et al. 2004). We introduced some simplifications of the test statistic using basic results in branching process.

Value

The function provides textual results and a P-value for the null hypothesis of no diversification rate shift against a (lambda2/lambda1)-fold shift at the node under consideration.

Author(s)

Eric Durand <eric.durand@imag.fr>
Olivier Francois <olivier.francois@imag.fr>
References


Examples

```r
## Detecting diversification rate variation in bird families (135 tips)
data(bird.families)
tree.birds <- as.treeshape(bird.families, model = "yule")
class(tree.birds) <- "treeshape"
pv <- sapply(1:135, FUN = function(i) shift.test(tree.birds, i, lambda1 = 1,
                       lambda2 = 100, nrep = 1000, silent = TRUE))

## Significant shifts detected at nodes = 67 and 78
pv[c(67,78)]
shift.test(tree.birds, node = 67, lambda1 = 1, lambda2 = 100, nrep = 10000, silent = TRUE)
shift.test(tree.birds, node = 78, lambda1 = 1, lambda2 = 100, nrep = 10000, silent = TRUE)

## visualize the shifts
par(mfrow=c(2,1))
plot(cutreeshape(tree.birds, ancestor(tree.birds, 67) , "bottom"))
plot(cutreeshape(tree.birds, 78 , "bottom"))
```

simulate.R

Simulate R knowing K=k

Description

Simulate a split conditioned to the number of marks in each resulting subinterval

Usage

`simulate.R(beta, n, K)`

Arguments

- `beta`: Imbalance index
- `n`: Tip number in the parent clade
- `K`: Tip number in the left daughter clade

Author(s)

Odile Maliet, Fanny Gascuel & Amaury Lambert
**simulate.R.K**

### References


---

### simulate.R.K

**Simulate (R,K)**

**Description**

Simulate a split and the number of marks in each resulting subintervals

**Usage**

```r
simulate.R.K(beta, n)
```

**Arguments**

- `beta`: Imbalance index
- `n`: Tip number in the parent clade

**Author(s)**

Odile Maliet, Fanny Gascuel & Amaury Lambert

**References**


---

### simulate.Tau.X

**Simulate (Tau, X_{Tau-})**

**Description**

Simulates the time before the splitting of the marks into two different fragments, and the size of the fragment at this time.

**Usage**

```r
simulate.Tau.X(epsilon, x, alpha, beta, n, ab = FALSE, eta = 1, x.ab = 1, lambda = NULL)
```
simulate.Yi

Simulates the random variables $Y_i$

**Arguments**

- `N` Number of simulated $Y_i$
- `epsilon` Minimum size of unsampled splits (see appendix 1)
- `beta` Imbalance index
- `n` Number of marks in the interval

**Details**

$(Y_i)$ are independent random variables with density $\frac{2}{(\lambda_\epsilon)}\exp(-\beta n + 1)\exp(-x)(1 - \exp(-x))^\beta$

**References**

**simulate_kingman**

**Author(s)**
Odile Maliet, Fanny Gascuel & Amaury Lambert

**References**

---

**simulate_kingman**  
*Ranked topology with Kingman’s coalescent depths*

**Description**
Simulates the ranked topology with node depths simulated using the Kingman’s coalescent model

**Usage**

```r
simulate_kingman(epsilon, alpha, beta, N, equal.ab = TRUE, eta = 1, lambda = NULL)
```

**Arguments**
- `epsilon` Minimum size of unsampled splits (see appendix 1)
- `alpha` Clade age-richness index
- `beta` Imbalance index
- `N` Tip number
- `equal.ab` If set to TRUE, all species have the same probability to go extinct first (default to TRUE)
- `eta` Clade abundance-richness index (if equal.ab == FALSE)
- `lambda` Optional, vector of lambda_0 returned by the function lambda_0. If set to NULL (the default) it will be computed inside the function

**Value**
A phylo object with ranked shape drawn from our model, with an additional tip.ab field containing a vector of tip abundances.

**Author(s)**
Odile Maliet, Fanny Gascuel & Amaury Lambert

**References**
Examples

```r
# Simulate a tree
set.seed(813)
tree=simulate_kingman(epsilon=0.001, alpha=-1, beta=-1, N=20, equal.ab=FALSE, eta=1.5)

# Plot the tree with dots at tips that have sizes scaling with log abundance
tree$tip.label = rep(".", length(tree$tip.label))
plot.phylo(tree, show.node.label=TRUE,
  cex=(log(tree$tip.ab)-min(log(tree$tip.ab)-0.1))*
  6/diff(range(log(tree$tip.ab))), adj=0.1)
```

simulate_tree  Simulates ranked topology

Description

Simulates a ranked topology with tip abundance data from the beta alpha eta model

Usage

```r
simulate_tree(epsilon, alpha, beta, N, equal.ab = TRUE, eta = 1, lambda = NULL)
```

Arguments

- **epsilon**: Minimum size of unsampled splits (see appendix 1)
- **alpha**: Clade age-richness index
- **beta**: Imbalance index
- **N**: Tip number
- **equal.ab**: If set to TRUE, all species have the same probability to go extinct first (default to TRUE)
- **eta**: Clade abundance-richness index (if equal.ab == FALSE)
- **lambda**: Optional, vector of lambda_epsilon returned by the function lambda_N. If set to NULL (the default) it will be computed inside the function

Value

A phylo object with ranked shape drawn from our model, with an additional tip.ab field containing a vector of tip abundances.

Branch lengths are so that node depths are in 1:(n-1)

Author(s)

Odile Maliet, Fanny Gascuel & Amaury Lambert
simulate_yule

Reference


Examples

```r
# Simulate a tree
set.seed(813)
tree=simulate_tree(epsilon=0.001, alpha=2, beta=-1, N=20, equal.ab=FALSE, eta=0.5)

# Plot the tree with dots at tips that have sizes scaling with log abundance
tree$tip.label = rep(".", length(tree$tip.label))
plot.phylo(tree, show.node.label=TRUE,
cex=(log(tree$tip.ab)-min(log(tree$tip.ab)-0.1))*
6/diff(range(log(tree$tip.ab))), adj=0.1)
```

simulate_yule

Ranked topology with Birth-death process depths

Description

Simulates the ranked topology with node depths simulated using the constant birth-death process

Usage

```r
simulate_yule(epsilon, alpha, beta, N, b, d,
tmax = Inf, equal.ab = TRUE, eta = 1, lambda = NULL)
```

Arguments

- **epsilon**: Minimum size of unsampled splits (see appendix 1)
- **alpha**: Clade age-richness index
- **beta**: Imbalance index
- **N**: Tip number
- **b**: Birth rate
- **d**: Death rate
- **tmax**: Maximal crown age
- **equal.ab**: If set to TRUE, all species have the same probability to go extinct first (default to TRUE)
- **eta**: Clade abundance-richness index (if equal.ab == FALSE)
- **lambda**: Optional, vector of lambda_epsilon returned by the function lambda_N. If set to NULL (the default) it will be computed inside the function
smaller.clade.spectrum

Value

A phylo object with ranked shape drawn from our model, with an additional tip.ab field containing a vector of tip abundances.

Author(s)

Odile Maliet, Fanny Gascuel & Amaury Lambert

References


Examples

# Simulate a tree
set.seed(813)
tree=simulate_yule(epsilon=0.001, alpha=-1, beta=-1,
N=20, equal.ab=FALSE, eta=1.5, b=1, d=0.5)

# Plot the tree with dots at tips that have sizes scaling with log abundance
tree$tip.label = rep(".", length(tree$tip.label))
plot.phylo(tree, show.node.label=TRUE,
cex=(log(tree$tip.ab)-min(log(tree$tip.ab)-0.1))*6/diff(range(log(tree$tip.ab))), adj=0.1)

smaller.clade.spectrum(tree)

Compute the smaller clade spectrum of a tree.

Description

smaller.clade.spectrum returns a n*2 matrix where n is the number of internal nodes of the tree. For each i in 1:n, the [i,1] element of the matrix is the size of the clade rooted at the 'i'th node of the tree. [i,2] is the size of the smaller daughter clade at the root.

Usage

smaller.clade.spectrum(tree)

Arguments

tree An object of class "treeshape".

Value

A n*2 matrix (where n is the number of internal nodes of the tree) containing the size of the clades and the smaller clades. smaller.clade.spectrum(tree)[1,1] contains the number of tips of the tree. smaller.clade.spectrum(tree)[i,1] contains the number of tips of the subtree whose root is the node number n-i+1. smaller.clade.spectrum(tree)[1,2] contains the number of tips of the smaller daughter clade at the root.
spectrum.treeshape

Author(s)

Michael Blum <michael.blum@imag.fr>
Nicolas Bortolussi <nicolas.bortolussi@imag.fr>
Eric Durand <eric.durand@imag.fr>
Olivier Francois <olivier.franois@imag.fr>

See Also

spectrum.treeshape

Examples

# computes the log-likelihood for Aldous' model
shape.new <- function(tree){
  h <- function(n){sum(1/(1:n))}
  mat <- smaller.clade.spectrum(tree)
  parent <- mat[,1]
  daughter <- mat[,2]
  nh.n <- sapply(parent, FUN = function(x){x*h(x-1)})
  s <- sum(log(daughter/parent) + log(1 - daughter/parent) + log(nh.n))
  return(s)}

# distribution over 200 replicates
tr <- rtreeshape(200, 100, FUN = function(n,i){if((i>0)&(i<n))return(1/i/(n-i)) else return(0)})
res <- sapply( tr, FUN = shape.new)
hist(res)

spectrum.treeshape  Compute the spectrum of a tree

Description

This function returns a sequence containing the number of subtrees of size n, n-1, ..., 3, 2 where n
is the size of the tree. The 'k'th element of the sequence is the number of subtrees of size n-k+1 in
the tree, where n is the number of tips of the tree.

Usage

spectrum.treeshape(tree)

Arguments

tree  An object of class "treeshape".
Value

A sequence of size n-1 (where n is the number of tips of the tree) containing the number of subtrees of each size. spectrum.treeshape(tree)[1] is the number of subtrees with n tips (equal to 1). spectrum.treeshape(tree)[n-1] is the number of cherries of the tree (subtrees with 2 tips).

Author(s)

Michael Blum <<michael.blum@imag.fr>>
Nicolas Bortolussi <<nicolas.bortolussi@imag.fr>>
Eric Durand <<eric.durand@imag.fr>>
Olivier Francois <<olivier.francois@imag.fr>>

See Also

smaller.clade.spectrum

Examples

```R
## A random Yule tree with 30 tips
tr<-rtreeshape(n=1,tip.number=30,model="yule")
tr<-tr[[1]]
spectre=spectrum.treeshape(tr)
spectre

## Number of cherries of the tree : nrow(tr$merge)==29
spectre[29]
```

### split

**Computes the splits at each node of a tree**

Description

Computes the splits at each node of a tree

Usage

`split(tree)`

Arguments

- `tree`: A phylo object

Author(s)

Odile Maliet, Fanny Gascuel & Amaury Lambert
**subtree.test**  
*Test the Yule or PDA hypothesis*

**Description**
subtree.test tests the likelihood of the Yule or the PDA hypothesis and computes the p-value of the test. The test is based on the number of subtrees of a given size in the tree.

**Usage**
subtree.test(tree, size, alternative = "two.sided")

**Arguments**
- **tree**: An object of class "treeshape".
- **size**: The size of the subtrees to test for.
- **alternative**: The alternative hypothesis of the test. It can be "two.sided" (default), "less" or "greater".

**Details**
See references for the mathematical details of the test. It uses a Gaussian approximation to compute the p-value.

**Value**
A list containing the following arguments:
- **statistic**: the value of the statistic used in the test
- **p.value**: the p-value of the test
- **alternative**: the alternative hypothesis used for the test

**Author(s)**
Michael Blum <michael.blum@imag.fr>
Nicolas Bortolussi <nicolas.bortolussi@imag.fr>
Eric Durand <eric.durand@imag.fr>
Olivier Francois <olivier.francois@imag.fr>

**References**

**See Also**
sackin.test  
colless.test
Examples

```r
## Generate a random pda tree with 50 tips
tr<-rtreeshape(n=1,tip.number=50,model="pda")
tr<-tr[[1]]

## Test the yule hypothesis, using subtrees of size 2 (Cherries),
## with the alternative hypothesis "less"
subtree.test(tr,size=2,alternative="less")
```

---

**summary.treeshape**

Print a summary of an object of class "treeshape"

### Description

This function prints a compact summary of a phylogenetic tree of class "treeshape"

### Usage

```r
## S3 method for class 'treeshape'
summary(object, ...)
```

### Arguments

- `object` an object of class "treeshape".
- `...` further arguments passed to or from other methods.

### Details

`summary.treeshape` prints the following information: the number of tips of the tree, its Colless' index, and the expected values and standard deviations of the Colless' index under the PDA and Yule models. The expected value of the Colless' index under the Yule model is given according to the formula: \( n \times \log n + n \times (\gamma - 1 - \log 2) \) where \( n \) is the number of tips of the tree and \( \gamma \) the Euler's constant. The standard deviation under the Yule model is given by: \( \sqrt{3 - \frac{\pi^2}{6} - \log 2} \times n \).

The expected value of the Colless' index under the PDA model is given according to the formula: \( \sqrt{\pi} \times n^{3/2} \). The standard deviation under the PDA model is given by: \( \sqrt{\frac{10}{3} - \pi \times n^{3/2}} \).

### Value

A NULL value is returned, the results are simply printed.

### Author(s)

Michael Blum <michael.blum@imag.fr>
Nicolas Bortolussi <nicolas.bortolussi@imag.fr>
Eric Durand <eric.durand@imag.fr>
Olivier Francois <olivier.francois@imag.fr>
See Also

summary
colless for more informations about the expected values under the Yule and PDA models.

Examples

```r
## Summary of a PDA tree with 100 tips.
summary(rpda(100))
## Note that the standard deviation is very large.

## Summary of a Yule tree with 100 tips.
summary(ryule(100))
## The standard deviation under the Yule model is much smaller than under
## the PDA model.

## Summary of the HIV tree.
data(hivtree.treeshape)
summary(hivtree.treeshape)
## The HIV tree is much closer from the Yule model than from the PDA model.
```

###叮叮
tipsubtree

**Extract a subtree that contains pre-specified tip names or labels**

**Description**

tipsubtree returns an object of class "treeshape" that contains pre-specified tip names or labels. The name of the tips are conserved. It extracts the smallest tree that contains the common ancestors of the given tips.

**Usage**

tipsubtree(tree, tips, numeric=FALSE)

**Arguments**

tree An object of class "treeshape".
tips A vector that contains the names of the tips one want to keep. Warning, the names are case-sensitive.
numeric An object of class "logical". If FALSE, the vector tips contains the names of the tips. If TRUE, it contains the labels of the tips.

**Value**

An object of class "treeshape"
transform

Internal function

Description

Write the tree in another form, used in the mcmc_alpha function

Usage

transform(tree, unsampled, list_depth = NULL, change_depth = 0)

Arguments

tree
unsampled
list_depth
change_depth
transform_eta

Author(s)
Odile Maliet, Fanny Gascuel & Amaury Lambert

transform_eta  Internal function

Description
Write the tree in another form, used in the mcmc_eta function

Usage
transform_eta(tree, unsampled, eta, list_depth = NULL, change_depth = 0)

Arguments
tree
unsampled
eta
list_depth
change_depth

Author(s)
Odile Maliet, Fanny Gascuel & Amaury Lambert

treeshape  Builds an object of class treeshape

Description
treeshape builds a tree of class "treeshape" from a n*2 matrix, where n is the number of internal nodes of the tree. There are no informations about the heights of the branches in an object of class "treeshape". Formally, a "tree shape" is a phylogenetic tree where the label of the tips are ignored. Here, the label of the tips can be kept or ignored. If a names vector is provided, then the names of species are attached to the tips of the tree. Otherwise, tips are simply labeled with their numbers in the tree. Trees of class "treeshape" are always binary.

Usage
treeshape(nodes, names)
Arguments

nodes nodes is a n*2 matrix containing the node structure of the tree.
names names is a vector which contains the names of the tips.

Details

A tree of class "treeshape" is a fully dichotomous binary tree. The purpose of the class "treeshape" is to study the topology of phylogenetic trees. The heights of branches are not provided for a tree of that class because we mainly focus on the balance aspect of the trees. The 'i'\textsuperscript{th} row of the nodes matrix represents the children of the node number i in the tree (nodes[i,1] being the left child, and nodes[i,2] being the right child). A positive value represents an internal node, while a negative one stands for a tip of the tree. The last row always represents the children of the root of the tree.

Value

An object of class "treeshape"

Author(s)

Michael Blum <<michael.blum@imag.fr>>
Nicolas Bortolussi <<nicolas.bortolussi@imag.fr>>
Eric Durand <<eric.durand@imag.fr>>
Olivier Francois <<olivier.francois@imag.fr>>

References


See Also

rtreeshape

Examples

## Nodes will define the nodes of a five tips tree
nodes<-matrix(nrow=4,ncol=2)
nodes[1,]<-c(-5,-4)
nodes[2,]<-c(1,-1)
nodes[3,]<-c(-3,2)
nodes[4,]<-c(-2,3)

## Now we can build the tree and plot it.
tree1<-treeshape(nodes)
plot(tree1)

## Computation of the sackin index for the tree :
## universal.treeshape

sackin(tree1)

### Label will define the names of the tips
label=c("a", "b", "c", "d", "e")
tree2<-treeshape(nodes, label)
plot(tree1, tree2)

---

**universal.treeshape**  
Universal phylogenetic tree of life

### Description

This data set describes the Universal Tree of Life described by CR Woese, in the `treeshape` class. "The universal phylogenetic tree not only spans all extant life, but its root and earliest branchings represent stages in the evolutionary process before modern cell types had come into being."

### Usage

```r
data(universal.treeshape)
```

### Format

universal.treeshape is an object of class "treeshape".

### References


### Examples

```r
## Example tree in "treeshape" format
data("universal.treeshape")

## Summary of the tree
summary(universal.treeshape)

## Visual representation of the tree
plot(universal.treeshape)
```
**yule_lengths**  
*Internal BetaAlphaEta function*

**Description**

Simulates node depths in the birth-death model, conditioned by the number of tips and the age of the root (using the expression in "The shape and probability of reconstructed phylogenies", by Amaury Lambert and Tanja Stadler (Theoretical Population Biology, 2013))

**Usage**

```r
yule_lengths(N, b, d, tmax)
```

**Arguments**

- **N**: Tip number
- **b**: Birth rate
- **d**: Death rate
- **tmax**: Maximum age of the root

**Author(s)**

Odile Maliet, Fanny Gascuel & Amaury Lambert

**References**

Index

* datagen
  rtreeshape, 36
  treeshape, 55
* datasets
  carnivora.treeshape, 10
cytochromc, 16
  hivtree.treeshape, 22
  primates, 35
  rhodopsin, 35
  universal.treeshape, 57
* hplot
  plot.treeshape, 34
* htest
  aldous.test, 3
  index.test, 22
  likelihood.test, 26
  maxlik.betasplit, 28
  shift.test, 41
  subtree.test, 51
* manip
  all.equal.treeshape, 4
  as.phylo.treeshape, 6
  as.treeshape, 7
  cutreeshape, 15
  summary.treeshape, 52
  tipsubtree, 53
* univar
  cladesize, 12
  colless, 13
  sackin, 38
  shape.statistic, 39
  smaller.clade.spectrum, 48
  spectrum.treeshape, 49

a_N, 8
aldous.test, 3
all.equal, 5
all.equal.treeshape, 4
as.phylo, 6
as.phylo.treeshape, 6, 7
as.treeshape, 6, 7
auxlik, 8
bind.trees, 9
build_tree, 10
carnivora.treeshape, 10
change_int, 11
change_int_eta, 11
colless, 13, 24, 29, 39, 53
colless.test, 29, 51
colless.test(index.test), 22
cutreeshape, 15, 54
cytochromc, 16
depth, 16
enhance, 17
enhance_eta, 17
get_extinction_list, 18
get_PD_sample, 19
get_tree_beta, 20
hivtree.treeshape, 22
index.test, 22
insert, 24
lambda.epsilon, 25
lambda_N, 26
likelihood.test, 26
maxlik.betasplit, 28
mcmc_alpha, 30
mcmc_eta, 31
nodes_depths_ordonnes, 33
plot, 34
plot.treeshape, 34
primates, 35
rbactrian, 35
rhodopsin, 35
rtreeshape, 36, 56
sackin, 14, 24, 29, 38
sackin.test, 29, 39, 51
sackin.test (index.test), 22
shape.statistic, 28, 39
shift.test, 41
simulate.R, 42
simulate.R.K, 43
simulate.Tau.X, 43
simulate.Y1, 44
simulate_kingman, 45
simulate_tree, 46
simulate_yule, 47
smaller.clade.spectrum, 48, 50
spectrum.treeshape, 49, 49
split, 50
subtree.test, 24, 51
summary, 53
summary.treeshape, 52
tipsubtree, 16, 53
transform, 54
transform_eta, 55
treeshape, 55
universal.treeshape, 57
yule_lengths, 58