Package ‘phyloTop’

February 20, 2015

Version 1.1.1
Type Package
Title Phylogenetic Tree Topological Properties Evaluator
Date 2014-05-10
Author Michael Boyd and Caroline Colijn
Maintainer Michael Boyd <mboyd855@gmail.com>
Description Calculates some topological properties of phylogenetic trees
License GPL-2
Depends phylobase, NHPPoisson, ape, igraph
NeedsCompilation no
Repository CRAN
Date/Publication 2014-05-14 02:48:06

R topics documented:

  phyloTop-package .................................................. 2
  allNodeAnalysis .................................................. 2
  configurations .................................................. 4
  exampleSummary .................................................. 6
  fastModelAnalysis .............................................. 8
  homTrees ......................................................... 10
  laddItr .......................................................... 11
  modelAnalysis ................................................... 12
  nodeAnalysis .................................................... 14
  rootLaddDist .................................................... 15
  rTrees ........................................................... 16
  SimulatedOutbreakGenealogy .................................... 17
  ssTrees .......................................................... 18
  topFuncs .......................................................... 19
  treeAnalysis ...................................................... 21
  treeGraphics ..................................................... 24
  utility ............................................................ 25
Description

Calculates some topological properties of phylogenetic trees.

Details

<table>
<thead>
<tr>
<th>Package</th>
<th>phyloTop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Package</td>
</tr>
<tr>
<td>Version</td>
<td>1.1</td>
</tr>
<tr>
<td>Date</td>
<td>2013-10-05</td>
</tr>
<tr>
<td>License</td>
<td>GPL-2</td>
</tr>
</tbody>
</table>

This package is based on the package phylobase. See its documentation for many useful related functions. In particular nodeid and treewalk. `library(help=phyloTop)` will give more information on the package.

Many thanks to Caroline Colijn of Imperial College who provided some of the data accompanying the package and supervised me in the writing of it.

Note

I recommend running `phylobase.options(allow.duplicated.labels='ok')` to remove some warning messages.

Author(s)

Michael Boyd

Maintainer: Michael Boyd <mboyd855@gmail.com>

Description

A collection of functions designed to examine all of the nodes in a tree at once. Note that not all of these functions return topological properties of the trees - two trees could be topologically identical but give different results. Many of the functions below gives results organized by nodeid.


Usage

\[
\text{nDescendants(tree)} \\
\text{nTipDescendants(tree)} \\
\text{dists(tree)} \\
\text{ladderNums(tree)} \\
\text{treeImb(tree)} \\
\text{nodeApply(tree, func, showId = TRUE)}
\]

Arguments

\[
\text{tree} \quad \text{An object of class phylo4} \\
\text{func} \quad \text{A function with input being a binary phylogenetic tree and output being a number} \\
\text{showId} \quad \text{A Boolean}
\]

Details

\text{nDescendants} gives the number of descendants (including internal nodes) of all the nodes in the tree. It is organized by \text{nodeId}. Note that when \text{phylobase}'s \text{descendants} is given a tip it returns that tip as an output.

\text{nTipDescendants} gives the number of tip descendants of all the nodes in the tree. It is organized by \text{nodeId}.

\text{dists} gives the distance of each node from the root of the tree. It is organized by \text{nodeId}. Note that, in \text{phylobase}, \text{Depth} is how far the node is from the root taking edge length into account. I have stuck with this convention so \text{Dist} is the number of steps required to go to that node from the root.

\text{ladderNums} gives the largest ladder size (away from the root) starting from the given node. It is based on \text{rootLaddDist}. Note that it returns a vector of length equal to the number of internal nodes of the tree. Organized by \text{nodeId}. \text{laddItR} is intended to be applied to this result of this function.

\text{treeImb} gives the node imbalance of every internal node in the tree using \text{nodeImb}. The output ordered by \text{nodeId}.

\text{nodeApply} applies the a given function to all the non-trivial subtrees within the given tree. That is, for every internal node there is a tree - the tree descending from that node. \text{nodeApply} applies \text{func} to all of these trees. The output is ordered by \text{nodeId} and if \text{showId}=\text{TRUE} then the name of the output is set to show the \text{nodeId} of the node corresponding to each subtree.

Value

\text{nDescendants} \quad \text{An integer vector of length equal to the number of nodes in the tree.}

\text{nTipDescendants} \quad \text{An numeric vector of length equal to the number of nodes in the tree. All entries are integers.}

\text{ladderNums} \quad \text{A numeric vector of length equal to the number of internal nodes of the tree. All entries are integers.}
configurations

- **treeImb**: A matrix with two rows and number of columns equal to the number of internal nodes of the tree.
- **nodeApply**: A numeric vector of length equal to the number of internal nodes. If `showId=TRUE`, then the vector is named.

**Author(s)**

Michael Boyd

**See Also**

- `nodeId`, `rootLaddDist`. See `treeAnalysis` for more examples to use in `func`.

**Examples**

```r
## Not run:
## Creates a random tree of class phylo4
tree <- rtree4(50)

## Label the nodes of a tree by their nodeId
labelledTree <- idNodeLabel(tree)
plot(labelledTree,show.tip.label=FALSE,show.node.label=TRUE)

## Finds the number of descendants of the nodes
nDescendants(tree)

## Finds the number of tip descendants of the nodes in the tree
nTipDescendants(tree)

## Finds the distance of each node from the root of the tree
dists(tree)

## Finds the length of the internal ladders in the tree
ladderNums(tree)

## Calculates the imbalance of each of the internal nodes
treeImb(tree)

## This returns the Colless imbalance of all subtrees of the tree
## The vector is named by the nodeId of the root of each subtree
nodeApply(tree,colless)

## End(Not run)
```

**configurations**

*Configurations in Trees*
**configurations**

**Description**

A collection of functions for examining configurations in trees. A configuration of size \( n \) is a subtree with \( n \) tips. So a cherry (for example) is a 2-configuration. Note that not all of these functions return topological properties of the trees. For example, `configFind` can give very different results for two trees with the same topology.

**Usage**

```r
nConfig(tree, configSize)
configFind(tree, configSize)
configTop(tree)
classifyConfigFour(tree)
classifyConfigFive(tree)
```

**Arguments**

- `tree` An object of class `phylo4`
- `configSize` An integer specifying the size of configuration for which to search

**Details**

- `nConfig` (number of configurations) finds the number of configurations of the specified size in the tree. This function (and many of those listed below) use `nTipDescendants`.
- `configFind` returns a vector with each element being the `nodeId` of the root of a configuration of the specified size. That is, it gives all the nodes which have the specified number of tip descendants.
- `configTop` returns a vector giving the number of \( n \)-congfigurations for each \( n \) for which one could exist. If there are \( t \) tips then there is always one configuration of size \( t \) (the whole tree) and \( t \) configurations of size one (the tips). So the vector is length \( t \). Note that this is a topological property and is very useful for distinguishing trees with many tips.
- `classifyConfigFour` gives the number of each of the different types of configurations of size four. There are two such types one called `laddType` and the other `branchType`. There are many ways in which they are topologically different - this function uses the number of tip children of the root. `laddType` has only one while the other has none.
- `classifyConfigFour` works similarly but except it looks at configurations size 5. Here there are three types. It first separates by number of tip children of the root (one has none and is called `branchType`) and then by size of ladder from the root (the type with a longer ladder is `laddType` and the other is `fourType` as there is a 4-1 split at the top).

**Value**

- `nConfig` An integer - the number of configurations of the specified size in the tree.
- `configSize` An integer vector of length given by the number of configurations of the specified size.
- `configTop` An integer vector of length given by the number of tips of the tree.
- `classifyConfigFour` A data frame with one row and two columns with headers specifying which type is which.
classifyConfigFive

A data frame with one row and three columns with headers specifying which
type is which.

Author(s)

Michael Boyd

See Also

configShow for a function which plots the tree highlighting configurations of a given size.
treeAnalysis for more functions examining topological properties of trees.
ntipDescendants.

Examples

```r
## Creates a random tree of class phylo and plots it with nodes labelled by ID
tree <- rtree(50)
tree <- idNodeLabel(tree)
plot(tree,show.tip.label=FALSE,show.node.label=TRUE)

## Finds the number of cherries (also called 2-configurations) in the tree
nConfig(tree,2)

## Gives you the nodeID of the roots of the cherries
configFind(tree,2)

## Finds the complete configuration profile of the tree
configTop(tree)

## Classifies the configurations of size four...
classifyConfigFour(tree)

## ...and the configurations of size five
classifyConfigFive(tree)
```

Example Tree Summary

Description

An example of treeListSummary that has been used on the trees accompanying the package:
ssTrees, homTrees and rTrees. See the examples section for some interesting plots.

Usage

data(exampleSummary)
Format
A data frame with 100 observations on the following 7 variables (including the Type of the tree).

- **avgladder** a numeric vector
- **nLadders** a numeric vector
- **maxLadder** a numeric vector
- **sackin** a numeric vector
- **colless** a numeric vector
- **nCherries** a numeric vector
- **Type** a factor with levels hom and SS

Details
rTrees were created using rtree4 as the model for modelCreate. ssTrees and homTrees were created by models based on the spread of disease. They they are from a transmission model with only one pathogen lineage per host. The former had a fifth of the hosts seeding five times as many infections as the others, and while the latter was homogeneous - everyone was equally infectious. They all have 60 nodes or a little more. Those generated by rtree4 have exactly 61.

Source
homTrees and ssTrees were created by Caroline Colijn of Imperial College London. rTrees were created with the functions of this package. rtree4 was the model used.

See Also
modelAnalysis for functions to create similar summaries based on your own models.

Examples

```r
# Load data
data(exampleSummary)

# See the format of the data
head(exampleSummary)

# Find the correlations between the numeric vectors in the summary for the two types of trees
# This can be very useful. Observe the differences in correlations
homCor <- cor(exampleSummary[1:50,1:6])
ssCor <- cor(exampleSummary[51:100,1:6])
randCor <- cor(exampleSummary[101:150,1:6])

## Not run:
### I WILL NOW ILLUSTRATE SOME USES OF THE PACKAGE
### THESE METHODS USE LIBRARY ggplot2
### IT WILL NEED TO BE SEPERATELY INSTALLED IF YOU WISH TO USE IT

# Initialize plot
p <- ggplot(data=exampleSummary) + facet_wrap(~Type)
```
# A bar plot of nLadders with colors determined by maxLadder
p + geom_bar(aes(nLadders, fill = factor(maxLadder)), binwidth = 1) +
guides(fill = guide_legend(title = 'maxLadder'))

# A scatter plot of maxLadder against avgLadder
p + geom_point(aes(maxLadder, avgLadder, color = Type)) + theme(legend.position = 'none')

## End (Not run)

fastModelAnalysis

---

**fastModelAnalysis**  
*Fast Model Analysis*

## Description

A collection of functions which calculate properties of trees more quickly. These functions do a similar job to those in modelAnalysis but they may be slightly more difficult to use and adapt.

Many functions in this package share preliminary calculations. This means that in using topSumm, for example, the same code is run several times. The idea behind these functions is to improve the speed by doing these calculations just once. These preliminary computations result in what I call a base. This will mostly be done with baseListCreate and is often most of the work in the calculation of a topology profile. Then, the final topologies of the given trees are created using baseListAnalysis. I refer to functions which create a base from a tree as baseFuncs and those which create a topology from a base as topFuncs.

## Usage

```r
baseCreate(tree, baseFuncs)
baseListCreate(treeList, baseFuncs, loadingBar = TRUE)
baseAnalysis(base, topFuncs)
baseListAnalysis(baseList, topFuncs, loadingBar = FALSE)
```

## Arguments

- **tree**: An object of class phylo4.
- **baseFuncs**: A list of functions which each take a phylogenetic tree as an input and give any output.
- **treeList**: A list where the elements of the list are objects of class phylo4.
- **loadingBar**: A Boolean. If true, it results in the number of elements evaluated so far being printed as each is evaluated. This gives some indication of how long the function will take.
- **base**: A list. Each element of the list is intended to be the result of an element of BaseFuncs on a phylogenetic tree.
- **baseList**: A list where each element is a base. This results in a nested list.
- **topFuncs**: A list of functions which each takes an element of the base and returns a number.
Details

baseCreate creates the base for a tree. That is, it runs a number of functions (contained in list baseFuncs) on the tree and stores the result in a list.

baseListCreate does the same as baseCreate but for a list of trees (called treeList). The result is therefore a list of bases. This is quite a complicated object. It is a list of the same length as treeList. Each element of this list is a base corresponding to a tree in treeList - that is a list of the same length as baseFuncs. loadingBar defaults to TRUE as this function may take a long time

baseAnalysis analyses a given base using functions (contained in topFuncs) which act on each member of the base. That is, the first element of topFuncs will take the first element of the base and return a number. The list of all these numbers is a topological profile of the tree. This list is what baseAnalysis returns.

baseListAnalysis does the same as baseList but to each element of a list of bases. The result is a dataframe where each row corresponds to a different base and each column to a different topological property.

Value

- baseCreate: A base. That is a list where each element is a result of a function on a tree. The length of the list will be the same as the length as baseFuncs
- baseListCreate: A list where each element is the base (for the given baseFuncs) for an element of treeList. So it's length will be the same as that of treeList
- baseAnalysis: A matrix with one row and number of columns equal to the length of topFuncs.
- baseListAnalysis: A dataframe where each row corresponds to a different base and each column to a different topological property. So the number of columns is equal to the length of topFunc and the number of rows is equal to the length of baseList.

Author(s)

Michael Boyd

See Also

- modelAnalysis contains simpler but slower functions with similar effects. There are also a number of ways to create treeList in this section.
- See allNodeAnalysis, configurations and treeAnalysis for some possibilities for what to put in baseFuncs.
- topFuncs conatins a number of useful options for putting in topFuncs

Examples

```r
## This is an example of using bases to speed up calculation of topological properties.

## Creates a tree
tree <- rtree4(50)

## Sets up the list of bases
basefuncs <- c(ladderNums, treeImb, nTipDescendants)

## Creates the base
base <- baseCreate(tree, basefuncs)

## Sets up the list of functions from the base
## Note that the functions are matched with the correct base element
topfuncs <- c(fNLadders, fColless, function(x) {fNConfig(x, 2)})

## Finds the topological properties
baseAnalysis(base, topfuncs)

## Not run:
## Now a new list of topfuncs
## Some of these use the same elements of the base
## This means that there will be a speed improvement


<table>
<thead>
<tr>
<th>homTrees</th>
<th>Homogeneous Trees</th>
</tr>
</thead>
</table>

**Description**

A list of 50 trees generated by a homogeneous model.

**Usage**

data(homTrees)

**Format**

A list of length 50 where each object is an object of class phylo4.

**Details**

ssTrees and homTrees were created by models based on the spread of disease. They they are from a transmission model with only one pathogen lineage per host. The former had a fifth of the hosts seeding five times as many infections as the others, and while the latter was homogeneous - everyone was equally infectious. They all have 60 nodes or a little more.
laddItr

Source

homTrees was created by Caroline Colijn of Imperial College London.

See Also

dotExample for a more detailed summary.

Examples

data(homTrees)
  ## Not run:
  toplist <- c(avgLadder, colless, function(tree){nConfig(tree, 2)})
  treeListSummary(homTrees, toplist)
  ## End(Not run)

laddItr  Ladder Iteration Tool

Description

A function for internal use in relation to calculating the number of ladders in a tree.

Usage

laddItr(ladderNumbers)

Arguments

ladderNumbers A numeric vector which is a result of calling ladderNums to a tree.

Details

This function reduces the maximum value of ladderNumbers by one. Say \( m \) is the maximum number and there are \( n \) of them. Then this function removes them and also \( n \) of the natural numbers up to \( m \) - essentially removing all the ladders of size \( m \) from the tree.

Value

Another numeric vector suitable for passing to laddItr.

Author(s)

Michael Boyd

See Also

dotExample and avgLadder which use this function.
Examples

```r
## Creates a tree
tree <- rtree4(50)

## Creates an appropriate vector
laddernumbers <- ladderNums(tree)

## Performs the operation
laddIttr(laddernumbers)
```

---

**Description**

A collection of functions to summarize the topological properties of trees generated by a model or other large groups of trees. Note that the examples section is quite detailed and many be useful to read. Some of them use the library ggplot2 which will need to be installed separately. More examples like this can be found in examplesummary.

**Usage**

```r
modelSummary(model, topList, n, loadingBar=TRUE)
modelCreate(model, n, loadingBar=FALSE)
treeListSummary(treeList, topList, loadingBar=TRUE)
```

**Arguments**

- `model`: A function with no inputs which results in an object of class phylo4. For example, `model = function() {rtree(50)}` is allowed. It is intended that there be some randomness in the model.
- `topList`: This is a list of functions. Each function in the list acts on an object of class phylo4 and returns a number - which is intended to be a topological property of the tree.
- `n`: An integer specifying the number of trees create.
- `loadingBar`: A Boolean. If true, it results in the number trees evaluated so far being printed as each tree is evaluated. This gives some indication of how long the function will take.
- `treeList`: A list where the elements of the list are objects of class phylo4.

**Details**

`modelSummary` generates some trees using the model created and returns a data frame with the topological summaries in `topList`. The rows correspond to the trees generated and the columns to the different summaries.
modelCreate creates a list of $n$ trees using the model specified. Note that loadingbar is FALSE by default here as most model's are likely to be quick.

treeListSummary performs the functions in topList on a list of trees. It returns a data frame where the rows correspond to the trees and the columns to the different summaries.

Value

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>modelSummary</td>
<td>A data frame with the rows corresponding to the different trees and the columns to the result of the topological summaries in topList</td>
</tr>
<tr>
<td>modelCreate</td>
<td>A list of $n$ objects of class phylo4</td>
</tr>
<tr>
<td>treeListSummary</td>
<td>A data frame with the rows corresponding to the different trees and the columns to the result of the topological summaries in topList</td>
</tr>
</tbody>
</table>

Author(s)

Michael Boyd

See Also

treeAnalysis and link{configurations} for some ideas of which functions to add to topList. exampleSummary for more examples of plots.

Examples

```r
## Create a function to test
maxLadd <- function(tree) {max(ladderNums(tree))}

## Create a model
## NOTE THE USE OF 'as'
bdModel <- function() {as(rlineage(0.5,0.1,Tmax=6),'phylo4')}

## Not run:
## A simple example of modelSummary
modelSummary(function(){rtree4(50)},topList=c(avgLadder,sackin),n=10)

## Another example using the created model and function
modelSummary(bdModel,topList=c(maxLadd,sackin),n=10,loadingBar=FALSE)

## Create a list of trees for more detailed analysis
bdTrees <- modelCreate(bdModel,10)

## Select a number of topological summaries to calculate
summaries <- c(avgLadder,maxLadd,sackin,function(tree){nConfig(tree,2)})

## Calculate these summaries for the trees in question
bdSummary <- treeListSummary(bdTrees,summaries)
names(bdSummary) <- c('avgLadder','maxLadd','sackin','Cherries')

### I WILL NOW ILLUSTRATE SOME USES OF THE PACKAGE
```
### Node Analysis

#### Description

A collection of functions which analyse the properties of particular nodes of the tree.

#### Usage

- `nodeImb(tree, node)`
- `ladderDist(tree, node)`
- `nodeDist(tree, node)`

#### Arguments

- `tree` An object of class `phylo4`
- `node` An integer specifying a node in a tree by its `nodeId`

#### Details

- `nodeImb` (stands for node imbalance) returns the number of tip descendants below the node on the left side and the right as a vector of size 2. `treeImb` calculates this for every node.
- `ladderDist` (ladder distance) gives the largest ladder size (away from the root) starting from the given node. It is based on `rootLaddDist`. Note that ladder size is given by the number of edges it is made up of.
- `nodeDist` gives the number of steps from the root to the given node. In phylebase, `Depth` is how far the node if from the root taking edge length into account. I have stuck with this convention so `Dist` is the number of steps required.
**rootLaddDist**

**Value**

- `nodeimb`: A numeric vector of size 2 (with integer values) giving the size (by number of nodes) of the trees descending from the node.
- `ladderDist`: A number - the length of the ladder from that node.
- `nodedist`: A number - the number of steps required to get from that node to the root.

**Author(s)**

Michael Boyd

**See Also**

`treeimb`, `rootLaddDist` and `nodeDepth`

**Examples**

```r
## Creates a random tree of class phylo4 and plots it with nodes labelled by ID
tree <- rtree4(50)
tree <- idNodeLabel(tree)
plot(tree, show.tip.label=FALSE, show.node.label=TRUE)

## Calculate the imbalance of a node
nodeimb(tree, 60)

## Calculates the length of the ladder from that node
ladderDist(tree, 60)

## Finds the number of steps required to get from that node to the root
nodedist(tree, 60)
```

---

**rootLaddDist**

*Root Ladder Distance*

**Description**

Gives the length of the longest ladder starting from the root of the tree.

**Usage**

`rootLaddDist(tree)`

**Arguments**

- `tree`: An object of class phylo4

**Details**

Used by `ladderDist`. Otherwise not intended to be used.
**Value**

A numeric vector of size one

**Author(s)**

Michael Boyd

**See Also**

ladderDist

**Examples**

```r
## Creates a random tree of class phylo4 and plots it with nodes labelled by ID
tree <- rtree4(50)
tree <- idNodeLabel(tree)
plot(tree,show.tip.label=FALSE,show.node.label=TRUE)

## Finds the length of the ladder staring from the root
rootLaddDist(tree)
```

---

### rTrees

**Random Trees**

**Description**

A list of 50 trees generated by rtree4.

**Usage**

```r
data(rTrees)
```

**Format**

A list of length 50 where each object is an object of class phylo4.

**Source**

Created by Michael Boyd.

**See Also**

exampleSummary for a more detailed summary.
### Description

A few functions to simulate a rather simple outbreak where the infectiousness is not homogeneous, and then to extract a genealogy for the simulated infections. The outbreak simulation uses the `simNHP.fun` function in the NHPoisson package. Functions `makephylotree` and `sortmyepi` are intended for use by `getgenealogy`, not direct use.

### Usage

```r
maketransintree(lambda, duration=1, NumCases=50)
getgenealogy(epirecord, epsilon=0.01)
```

### Arguments

- **lambda**: A vector specifying the rate of the non-homogeneous Poisson process over time (i.e., the infectivity). The size of lambda also specifies the resolution with which the infection process is simulated.
- **duration**: The duration of infectiousness (fixed).
- **NumCases**: An integer specifying the desired number of cases in the outbreak.
- **epirecord**: A simulated outbreak returned by `maketransintree`.
- **epsilon**: A small number to ensure that `getgenealogy` returns positive branch lengths. This is relevant when individuals recover/are sampled in the same time step in which they became infected or when individuals infect several others in the same time step.

### Details

`maketransintree` generates a transmission tree, starting with one infectious individual, where the infectiousness through time is given by the argument `lambda`.

`getgenealogy` extracts a genealogy from the simulated transmission tree returned by `maketransintree`. It assumes that branching points in the genealogy correspond to infection events in the outbreak and that cases are sampled at the time of recovery.
Value

- `maken transtree`: A matrix with 4 named columns: Infectee, Infector, InfTime, RecTime corresponding to the case number of the infected individual, the individual infecting them, the time of infection, and the individual’s recovery time (same as the sampling time).

- `getgenealogy`: An object of class phylo4 in which tips correspond to cases in the simulated outbreak and internal nodes correspond to the infection events among them.

Author(s)

Caroline Colijn

See Also

- `treeAnalysis` and `link{configurations}` for functions to explore the shapes of the genealogies.

Examples

```r
# specify an inhomogeneous rate of new infections
lambda <- c(rep(0.2, 10), rep(0.1, 10))  # twice as infectious in the first half of the infectious period, with a mean number of 3 secondary cases

## Create a simulated outbreak
epirecord <- maken transtree(lambda, duration=2, NumCases=30)

## Extract the genealogy as a phylo4 object
simgeneal <- getgenealogy(epirecord)

## Not run:
# Intermediate functions:
# creates the phylo4 object from a list of edges and lengths.
# makephylotree(Edges, Lengths, Root)

sortmyepi(epirecord)  # sorts the rows of the simulated epidemic

## End(Not run)
```

Description

A list of 50 trees generated by the super spreader model.

Usage

data(ssTrees)
Format

A list of length 50 where each object is an object of class phylo4.

Details

ssTrees and homTrees were created by models based on the spread of disease. They are from a transmission model with only one pathogen lineage per host. The former had a fifth of the hosts seeding five times as many infections as the others, and while the latter was homogeneous - everyone was equally infectious. They all have 60 nodes or a little more.

Source

ssTrees was created by Caroline Colijn of Imperial College London.

See Also

exampleSummary for a more detailed summary.

Examples

data(ssTrees)
## Not run:
toplist <- c(avgLadder, colless, function(tree)(nConfig(tree, 2)))
treeListSummary(ssTrees, toplist)
## End(Not run)
Arguments

 nodeDists  The result of running dists on a tree. That is, a vector organised by nodeId giving the distance of each node from the root.

 ladderNumbers  The result of running ladderNums on a tree. That is, a vector organised by nodeId giving the largest ladder size going away from the root starting from each node.

 treeImbalance  The result of running treeImb on a tree. See treeImb for more information.

 nTipDes  The result of running nTipDescendants on a tree. That is, a vector organised by nodeId giving the number of tip descendants of each node.

 configSize  An integer specifying the size of configuration to count.

Details

These functions all correspond to other functions based on trees. For example, fColless corresponds to colless. These functions are much faster but take inputs which already require a reasonable amount of work to calculate. To learn more about fColless see the documentation for colless.

For example, the intention is that when calculating many topological properties it may be faster to calculate ladderNumbers and nodeDists first and then run these functions.

Value

These functions all have the same value as their corresponding function.

Author(s)

Michael Boyd

See Also

fastModelAnalysis for more detailed examples.

Examples

```r
## Sets up an example tree
tree <- rtree4(50)

## This returns the same output as avgLadder(tree)
ladderNumbers <- ladderNums(tree)
fAvgLadder(ladderNumbers)

## But now we can calculate the number of ladders more efficiently.
fNLadders
```
**Tree Analysis Functions**

**Description**

A collection of functions to provide summary characteristics of phylogenetic trees. There are other functions which do this included in other documentation files (links below).

**Usage**

- `splitTop(tree, dist)`
- `sackin(tree)`
- `widths(tree)`
- `avgLadder(tree)`
- `nLadders(tree)`
- `colless(tree, normalize=TRUE)`
- `nodeFrac(tree, func, threshold)`
- `topSumm(tree, topList)`
- `cherries(tree)`
- `pitchforks(tree)`
- `maxheight(tree)`
- `stairs(tree)`
- `ILnumber(tree)`

**Arguments**

- **tree**: An object of class `phylo4`
- **dist**: An integer specifying the distance at which to do the splitting
- **normalize**: A boolean specifying whether to normalize the Colless Imbalance
- **func**: A function with input being a phylogenetic tree and output being a number.
- **threshold**: A number.
- **topList**: This is a list of functions. Each function in the list acts on an object of class `phylo4` and returns a number - which is intended to be a topological property of the tree.

**Details**

- `splitTop` gives the number of tip children of each of the nodes at the given distance from the node. Note that, in `phylobase`, *Depth* is how far the node is from the root taking edge length into account. I have stuck with this convention so *Dist* is the number of steps required to go to that node from the root. Returns an error if there are no nodes of the given distance. Note that it has been ordered to make it a topological property; if there is no order then trees with the same topology can give different results.

- `sackin` gives the sum of the distance of each of the tips from the root. This is a form of Sackin Imbalance. You may need to check that the definition you are using is the same as the one given here.
widths gives the number of nodes at each distance from the root. Note that the elements of the
returned vector give the lengths of the splitTop vectors for each distance. It uses dists.

avgLadder gives the average length of all the ladders in the tree. Uses internal function laddItr.
nLadders gives the number of distinct ladders in the tree. Also uses laddItr.
colless returns the normalised Colless Imbalance. That is the sum of all the node imbalances
divided by \( \frac{2}{(n-1)(n-2)} \) where \( n \) is the number of tips. See nodeImb for the calculation of the imbalance for particular nodes. There is also an option to return the unnormalized Colless Imbalance.

nodeFrac returns the fraction of nodes for which func(subtree)\(>=\)threshold where subtree is
the subtree descending from that node.

topSumm returns the result of each of the functions in topList on the specified tree. Similar functions (mostly applied to many trees or a model for generating trees) can be found in modelSummary.

cherries returns the number of cherries in the tree. A cherry is a node with two tip descendants.
pitchforks returns the number of pitchforks in the tree. A pitchfork is a node with one cherry descendant and one tip descendant.

maxheight returns the maximum height (discrete steps from the root, not taking branch length into account, or equivalently with branch lengths equal to 1) of any tip in the tree.


ILnumber returns the number of nodes with a single tip descendant in the tree.

Value

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>splitTop</td>
<td>An integer vector of length equal to the number of nodes of the tree of the given distance - it is their number of tip children. Note that taking split topologies at every level is a very descriptive topological property and is useful for testing if trees with many nodes have the same topology.</td>
</tr>
<tr>
<td>sackin</td>
<td>A numeric vector of length one giving the sum of the distance of each of the tips from the root.</td>
</tr>
<tr>
<td>widths</td>
<td>An integer vector of length equal to the largest distance of a tip to the root.</td>
</tr>
<tr>
<td>avgLadder</td>
<td>A number.</td>
</tr>
<tr>
<td>nLadders</td>
<td>A number.</td>
</tr>
<tr>
<td>colless</td>
<td>A number.</td>
</tr>
<tr>
<td>nodeFrac</td>
<td>A number.</td>
</tr>
<tr>
<td>topSumm</td>
<td>A numeric vector of the same length as topList.</td>
</tr>
<tr>
<td>cherries</td>
<td>A number.</td>
</tr>
<tr>
<td>pitchforks</td>
<td>A number.</td>
</tr>
<tr>
<td>maxheight</td>
<td>A number.</td>
</tr>
<tr>
<td>stairs</td>
<td>A vector whose first element is the average, over all internal nodes, of the absolute difference between the numbers of tips on the two edges descending from the node. The second element is the average of (number of tips in the smaller descending subtree)/(number of tips in the larger descending subtree).</td>
</tr>
<tr>
<td>ILnumber</td>
<td>A number.</td>
</tr>
</tbody>
</table>
Author(s)

Michael Boyd, Caroline Colijn

See Also

`modelSummary` for functions to produce data frames containing the results of several of these functions on many trees. There are also examples of `topSum` in the examples there.

`allNodeAnalysis` for more functions which give results about every node in the tree.

`configurations` for functions examining configurations in a tree. A cherry is a type of configuration (it is a 2-configuration).

`ladderShow` for a function which plots the tree highlighting the ladders.

`nodeApply` for a function related to `nodeFrac`.

`nodeImb`, `dists` and `laddItr`.

Examples

```r
## Creates a random tree of class phylo4 and plots it with nodes labelled by ID
tree <- rtree4(50)
tree <- idNodeLabel(tree)
plot(tree, show.tip.label=FALSE, show.node.label=TRUE)
## Not run:
## Finds the split topology of the fourth level
splitTop(tree, 4)

## Finds the Sackin Imbalance
sackin(tree)

## Finds the width topology of the tree
widths(tree)

## Finds the average ladder length
avgLadder(tree)

## Finds the number of distinct ladders in the tree
nLadders(tree)

## Finds the Colless Imbalance (normalized)
colless(tree)

## Finds the fraction of nodes for which colless(subtree of node) >= 0.25
nodeFrac(tree, colless, 0.25)

## End(Not run)
```
Description

A collection of functions to display features of trees graphically.

Note that simply typing (for example) ladderShow will show the code for that function - the code can be edited to display different features or add more options.

Usage

ladderShow(tree)
configShow(tree, configSize)
subtreeShow(tree, nodeList)

Arguments

tree
An object of class phylo4.

configSize
An integer specifying the size of configuration for which to highlight.

nodeList
A numeric vector specifying a subset of the nodes of the tree by their nodeid.

Details

ladderShow prints the tree with the ladders displayed in red.

configShow prints the tree with the configurations of the required size displayed in red.

subtreeShow prints the tree with the descendants of the nodes in nodeList highlighted. It is used in configShow and may be useful if you wish to write similar functions to highlight features of interest.

Value

These functions have no output - they simply print the tree.

Author(s)

Michael Boyd

See Also

cfgurations
Examples

```r
## Creates a random tree of class phylo4 and plots it with nodes labelled by ID
    tree <- rtree4(50)
    tree <- idNodeLabel(tree)
    plot(tree,show.tip.label=FALSE,show.node.label=TRUE)

## Plots the tree showing the ladders
    ladderShow(tree)

## Plots the tree showing the 3-configurations
    configShow(tree,3)

## Highlights the descendents of two particular nodes
    subtreeShow(tree,c(60,90))
```

---

utility  Utility Functions

Description

A collection of short, useful functions to save time and allow more intuitive use of the package.

Usage

```r
c4rtree(n)
c4rtreeNode(tree)
ntipchildren(tree,node)
c4rphyloCheck(object,error = TRUE)
```

Arguments

- `n` An integer
- `tree` An object of class phylo4
- `node` An integer specifying a node in a tree by it's nodeId
- `object` Any R object
- `error` a Boolean

Details

c4rtree uses ape function `rtree` and the as function of phylobase to produce a random tree of class phylo4 with a given number of tips. The algorithm used randomly splits one of the edges at each step.

c4rtreeNode labels the nodes of the tree by nodeId. Note that this includes the tips.
nTipChildren gives the number of children of a node which are tips.
c4rphyloCheck tests if a object is of the correct form for use in this package. It returns TRUE if the object is a binary tree of class phylo4 and FALSE (with an error message) if either of these conditions are not met. If error=FALSE then no error messages will be printed.
Value

- `rtree4` An object of class phylo4 with n tips.
- `idNodeLabel` An object of class phylo4 with nodes labelled by nodeId.
- `nTipChildren` An integer specifying the exact number of children of that node - note children not descendants.
- `phyloCheck` A Boolean specifying if the object is a tree suitable for use in this package.

Author(s)

Michael Boyd

See Also

- `rtree` and `nodeId`

Examples

```r
tree <- rtree(50)

labelledTree <- idNodeLabel(tree)
plot(labelledTree, show.tip.label=FALSE, show.node.label=TRUE)

nChildren <- nTipChildren(tree)

phyloCheck(tree)
```
Index

*Topic datasets
  exampleSummary, 6
  homTrees, 10
  rTrees, 16
  ssTrees, 18

allNodeAnalysis, 2, 9, 23
avgLadder, 11
avgLadder (treeAnalysis), 21

baseAnalysis (fastModelAnalysis), 8
baseCreate (fastModelAnalysis), 8
baseListAnalysis (fastModelAnalysis), 8
baseListCreate (fastModelAnalysis), 8

cherries (treeAnalysis), 21
classifyConfigFive (configurations), 4
classifyConfigFour (configurations), 4
colless (treeAnalysis), 21
configFind (configurations), 4
configShow, 6
configShow (treeGraphics), 24
configTop (configurations), 4
configurations, 4, 9, 23, 24
dists, 23
dists (allNodeAnalysis), 2

dedSummary, 6, 11, 13, 16, 19

fastModelAnalysis, 8, 20
fAvgLadder (topFuncs), 19
fColless (topFuncs), 19
fNConfig (topFuncs), 19
fNLadders (topFuncs), 19
fWidths (topFuncs), 19

getgenealogy
  (SimulatedOutbreakGenealogy), 17

homTrees, 10
idNodeLabel (utility), 25
ILNumber (treeAnalysis), 21

ladderDist, 16
ladderDist (nodeAnalysis), 14
ladderNums (allNodeAnalysis), 2
ladderShow, 23
ladderSort (treeGraphics), 24
laddItr, 11, 23

makephylotree
  (SimulatedOutbreakGenealogy), 17
maketranstree
  (SimulatedOutbreakGenealogy), 17

maxheight (treeAnalysis), 21
modelAnalysis, 7, 9, 12
modelCreate (modelAnalysis), 12
modelSummary, 23
modelSummary (modelAnalysis), 12

nConfig (configurations), 4
nDescendants (allNodeAnalysis), 2
nLadders, 11
nLadders (treeAnalysis), 21
nodeAnalysis, 14
nodeApply, 23
nodeApply (allNodeAnalysis), 2
nodeDepth, 15
nodeDist (nodeAnalysis), 14
nodeFrac (treeAnalysis), 21
nodeId, 4, 26
nodeImb, 23
nodeImb (nodeAnalysis), 14
nTipChildren (utility), 25
nTipDescendants, 6
nTipDescendants (allNodeAnalysis), 2

27
phyloCheck (utility), 25
phyloTop (phyloTop-package), 2
phyloTop-package, 2
pitchforks (treeAnalysis), 21

rootLaddDist, 4, 15, 15
rtree, 26
rtree4 (utility), 25
rTrees, 16

sackin (treeAnalysis), 21
SimulatedOutbreakGenealogy, 17
sortmyepi (SimulatedOutbreakGenealogy), 17
splitTop (treeAnalysis), 21
ssTrees, 18
stairs (treeAnalysis), 21
subtreeShow (treeGraphics), 24

topFuncs, 9, 19
topSumm (treeAnalysis), 21
treeAnalysis, 4, 6, 9, 13, 18, 21
treeGraphics, 24
treeImb, 15, 20
treeImb (allNodeAnalysis), 2
treeListSummary (modelAnalysis), 12

utility, 25

widths (treeAnalysis), 21