Package ‘phylobase’

February 20, 2015

Type Package
Title Base package for phylogenetic structures and comparative data
Version 0.6.8
Date 2014-03-20
Imports ade4, ape (>= 3.0), Rcpp (>= 0.11.0), methods
Depends grid
LinkingTo Rcpp
Suggests MASS, testthat (>= 0.8.1), RUnit
Author R Hackathon et al. (alphabetically: Ben Bolker, Marguerite Butler, Peter Cowan, Damien de Vienne, Dirk Eddelbuettel, Mark Holder, Thibaut Jombart, Steve Kembel, Francois Michonneau, David Orme, Brian O'Meara, Emmanuel Paradis, Jim Regetz, Derrick Zwickl)
Maintainer Francois Michonneau <francois.michonneau@gmail.com>
Description Provides a base S4 class for comparative methods, incorporating one or more trees and trait data
License GPL (>= 2)
Collate 'RcppExports.R' 'phylo4.R' 'checkdata.R' 'formatData.R'
  'class-multiphylo4.R' 'class-oldclasses.R' 'class-phylo4.R'
  'class-phylo4d.R' 'class-phylomats.R' 'methods-multiphylo4.R'
  'methods-oldclasses.R' 'methods-phylo4.R' 'methods-phylo4d.R'
  'setAs-Methods.R' 'pdata.R' 'subset.R' 'phylobase-package.R'
  'phylobase.options.R' 'prune.R' 'treePlot.R' 'treestruct.R'
  'treewalk.R' 'readNCL.R' 'bind.R' 'zzz.R'

URL http://phylobase.R-forge.R-project.org
NeedsCompilation yes
Repository CRAN
Date/Publication 2014-03-20 22:56:32
R topics documented:

- phylobase-package .............................................. 3
- addData .......................................................... 3
- as ................................................................. 5
- checkPhylo4 ...................................................... 6
- extractTree ....................................................... 8
- formatData ....................................................... 9
- geospiza .......................................................... 10
- getNode .......................................................... 11
- hasSingle ......................................................... 12
- Import Nexus and Newick files ............................... 13
- multiPhylo-class ............................................... 15
- nData ............................................................. 16
- owls4 ............................................................. 17
- pd.data ........................................................... 17
- pd.data-class .................................................... 18
- phylo4-accessors ............................................... 19
- phylo4-class ..................................................... 21
- phylo4-display ................................................. 21
- phylo4-labels .................................................... 23
- phylo4-methods .................................................. 26
- phylo4d .......................................................... 28
- phylo4d-class ................................................... 31
- phylo4d-display ................................................ 33
- phylo4d-hasData ............................................... 34
- phylobase.options ............................................. 35
- phylobubbles ................................................... 36
- phylog-class ..................................................... 37
- phylomat-class ................................................ 38
- phyloXXYY ...................................................... 39
- plotOneTree ..................................................... 40
- printphylo4 ..................................................... 41
- reorder-methods ............................................... 43
- subset-methods ................................................ 44
- tdata ............................................................. 47
- tip.data.plot ................................................... 48
- treePlot-methods .............................................. 49
- treewalk ........................................................ 51

Index 54
phylobase-package

Utilities and Tools for Phylogenetics

Description

Base package for phylogenetic structures and comparative data.

Details

- Package: phylobase
- Type: Package
- Date: 2009
- Depends: methods, grid, ape(>= 2.1)
- Suggests: adephylo, MASS
- License: GPL Version 2 or later
- Authors: R Hackathon et al. (alphabetically: Ben Bolker, Marguerite Butler, Peter Cowan, Damien de Vienne, Thibaut Jombart, Steve Kembel, Francois Michonneau, David Orme, Brian O’Meara, Emmanuel Paradis, Jim Regetz, Derrick Zwickl)
- URL: http://phylobase.r-forge.r-project.org/

MoreInfo

See the help index help(package="phylobase") and run vignette("phylobase", "phylobase") for further details and examples about how to use phylobase.

addData

Adding data to a phylo4 or a phylo4d object

Description

addData adds data to a phylo4 (converting it in a phylo4d object) or to a phylo4d object

Usage

```r
## S4 method for signature 'phylo4'
addData(x, tip.data, node.data, all.data,
  merge.data=TRUE, pos=c("after", "before"), ...)  
## S4 method for signature 'phylo4d'
addData(x, tip.data, node.data, all.data,
  merge.data=TRUE, pos=c("after", "before"), ...)  
```
Arguments

x a phylo4 or a phylo4d object

Tip.data a data frame (or object to be coerced to one) containing only tip data

Node.data a data frame (or object to be coerced to one) containing only node data

All.data a data frame (or object to be coerced to one) containing both tip and node data

Merge.data if both tip.data and node.data are provided, it determines whether columns with common names will be merged together (default TRUE). If FALSE, columns with common names will be preserved separately, with ".tip" and ".node" appended to the names. This argument has no effect if tip.data and node.data have no column names in common.

Pos should the new data provided be bound before or after the pre-existing data?

... additional arguments to be passed to formatData

Details

Rules for matching data to tree nodes are identical to those used by the phylo4d constructor.

If any column names in the original data are the same as columns in the new data, ".old" is appended to the former column names and ".new" is appended to the new column names.

The option pos is ignored (silently) if x is a phylo4 object. It is provided for compatibility reasons.

Value

addData returns a phylo4d object.

Author(s)

Francois Michonneau

See Also

tdata for extracting or updating data and phylo4d constructor.

Examples

data(geospiza)
nDt <- data.frame(a=rnorm(nNodes(geospiza)), b=1:nNodes(geospiza), row.names=nodeId(geospiza, "internal"))
t1 <- addData(geospiza, node.data=nDt)
Converting between phylo4/phylo4d and other phylogenetic tree formats

Description
Translation functions to convert between phylobase objects (phylo4 or phylo4d), and objects used by other comparative methods packages in R: ape objects (phylo, multiPhylo), ade4 objects (phylog, now deprecated), and to data.frame representation.

Usage

as(object, class)

Arguments

object a tree of class phylo4, phylo or phylog, or tree and data object of class phylo4d.
class the name of the class to which tree should be coerced (e.g., "phylo4" or "data.frame").

Methods

Coerce from one object class to another using as(object, "class"), where the object is of the old class and the returned object is of the new class "class". The as function examines the class of object and the new "class" specified to choose the proper conversion without additional information from the user. Conversions exist for combinations:

phylobase to phylobase formats:

as(object, "phylo4d") where object is of class phylo4 and returns an object of class phylo4d, with empty data.

as(object, "phylo") where object is of class phylo4d and returns an object of class phylo4. If data are dropped during the conversion, a warning message is produced. A similar conversion can be done by using the function extractTree, but in this case, no error message is produced.

phylobase to ape formats:

as(object, "phylo") where object is of class phylo4 or phylo4d and returns an object of class phylo. If data are dropped during the conversion from a phylo4d object, a warning message is produced.

ape to phylobase formats:

as(object, "phylo4") where object is of class phylo and returns an object of class phylo4.
as(object, "phylo4d") where object is of class phylo and returns an object of class phylo4d, with empty data.
as(object, "multiPhylo") --Not implemented yet. where object is of class multiPhylo and returns an object of class multiPhylo4.
checkPhylo4

phylobase to ade4 formats:

as(object, "phylo") where object is of class phylo4 and returns an object of class phylo.

Note that this format is now deprecated; the ade4 developers recommend that you use adephylo instead, which uses phylo and phylo4 formats natively.

phylobase format to data.frame:

as(object, "data.frame") where object is of class phylo or phylo4d and returns an object of class data.frame, with data included in the case of phylo4d.

Author(s)

Ben Bolker, Thibaut Jombart, Marguerite Butler, Steve Kembel

See Also

generic as, phylo4, phylo4d, extractTree, the original phylog from the ade4 package and as.phylo from the ape package.

Examples

```
tree.owls <- ape::read.tree(text=trString)
## round trip conversion
tree_in_phylo <- tree.owls  # tree is a phylo object
(tree_in_phylo4 <- as(tree.owls,"phylo4"))  # phylo converted to phylo4
identical(tree_in_phylo,as(tree_in_phylo4,"phylo"))
## test if phylo, and phylo4 converted to phylo are identical
## (no, because of dimnames)

## Conversion to phylog (ade4)
as(tree_in_phylo4, "phylog")

## Conversion to data.frame
as(tree_in_phylo4, "data.frame")

## Conversion to phylo (ape)
as(tree_in_phylo4, "phylo")

## Conversion to phylo4d, (data slots empty)
as(tree_in_phylo4, "phylo4d")
```

checkPhylo4

Validity checking for phylo4 objects

Description

Basic checks on the validity of S4 phylogenetic objects
Usage

checkPhylo4(object)
checkTree(object)
checkPhylo4Data(object)

Arguments

object A prospective phylo4 or phylo4d object

Value

As required by validObject, returns an error string (describing problems) or TRUE if everything is OK.

Note

These functions are only intended to be called by other phylobase functions.

checkPhylo4 is an (inflexible) wrapper for checkTree. The rules for phylo4 objects essentially follow those for phylo objects from the ape package, which are in turn defined in http://ape.mpl.ird.fr/misc/FormatTreeR_28July2008.pdf. These are essentially that:

• if the tree has edge lengths defined, the number of edge lengths must match the number of edges;
• the number of tip labels must match the number of tips;
• in a tree with ntips tips and nnodes (total) nodes, nodes 1 to ntips must be tips
• if the tree is rooted, the root must be node number ntips+1 and the root node must be the first row of the edge matrix
• tip labels, node labels, edge labels, edge lengths must have proper internal names (i.e. internal names that match the node numbers they document)
• tip and node labels must be unique

You can alter some of the default options by using the function phylobase.options.

For phylo4d objects, checkTree also calls checkPhylo4Data to check the validity of the data associated with the tree. It ensures that (1) the data associated with the tree have the correct dimensions, (2) that the row names for the data are correct.

Author(s)

Ben Bolker, Steven Kembel, Francois Michonneau

See Also

the phylo4 constructor and phylo4 class; formatData, the phylo4d constructor and the phylo4d class do checks for the data associated with trees. See coerce-methods for translation functions and phylobase.options to change some of the default options of the validator.
**extractTree**

*Get tree from tree+data object*

### Description

Extracts a phylo4 tree object from a phylo4d tree+data object.

### Usage

```r
extractTree(from)
```

### Arguments

- `from`: a phylo4d object, containing a phylogenetic tree plus associated phenotypic data. Created by the phylo4d() function.

### Details

`extractTree` extracts just the phylogeny from a tree+data object. The phylogeny contains the topology (how the nodes are linked together), the branch lengths (if any), and any tip and/or node labels. This may be useful for extracting a tree from a phylo4d object, and associating with another phenotypic dataset, or to convert the tree to another format.

### Author(s)

Ben Bolker

### See Also

- `phylo4`, `phylo4d`, `coerce-methods` for translation functions.

### Examples

```r
tree.phylo <- ape::read.tree(text = "((a,b),c);")
tree <- as(tree.phylo, "phylo4")
plot(tree)
tip.data <- data.frame(size = c(1, 2, 3), row.names = c("a", "b", "c"))
treedata <- phylo4d(tree, tip.data)
plot(treedata)
tree1 <- extractTree(treedata)
plot(tree1)
```
Description

Associates data with tree nodes and applies consistent formatting rules.

Usage

```r
formatData(phy, dt, type=c("tip", "internal", "all"),
    match.data=TRUE, rownamesAsLabels=FALSE,
    label.type=c("rownames", "column"),
    label.column=1, missing.data=c("fail", "warn", "OK"),
    extra.data=c("warn", "OK", "fail"), keep.all=TRUE)
```

Arguments

- **phy**: a valid phylo4 object
- **dt**: a data frame, matrix, vector, or factor
- **type**: type of data to attach
- **match.data**: (logical) should the row names of the data frame be used to be matched against tip and internal node identifiers? See details.
- **rownamesAsLabels**: (logical), should the row names of the data provided be matched only to labels (TRUE), or should any number-like row names be matched to node numbers (FALSE and default)
- **label.type**: character, rownames or column: should the labels be taken from the row names of `dt` or from the `label.column` column of `dt`?
- **label.column**: if `label.type` is "column", column specifier (number or name) of the column containing tip labels
- **missing.data**: action to take if there are missing data or if there are data labels that don’t match
- **extra.data**: action to take if there are extra data or if there are labels that don’t match
- **keep.all**: (logical), should the returned data have rows for all nodes (with NA values for internal rows when `type='tip'`, and vice versa) (TRUE and default) or only rows corresponding to the type argument

Details

`formatData` is an internal function that should not be called directly by the user. It is used to format data provided by the user before associating it with a tree, and is called internally by the phylo4d, tdata, and addData methods. However, users may pass additional arguments to these methods in order to control how the data are matched to nodes.

Rules for matching rows of data to tree nodes are determined jointly by the `match.data` and `rownamesAsLabels` arguments. If `match.data` is TRUE, data frame rows will be matched exclusively against tip and node labels if `rownamesAsLabels` is also TRUE, whereas any all-digit row...
names will be matched against tip and node numbers if rownamesAsLabels is FALSE (the default). If match.data is FALSE, rownamesAsLabels has no effect, and row matching is purely positional with respect to the order returned by nodeId(phy, type).

formatData (1) converts labels provided in the data into node numbers, (2) makes sure that the data are appropriately matched against tip and/or internal nodes, (3) checks for differences between data and tree, (4) creates a data frame with the correct dimensions given a tree.

Value

formatData returns a data frame having node numbers as row names. The data frame is also formatted to have the correct dimension given the phylo4 object provided.

Author(s)

Francois Michonneau

See Also

the phylo4d constructor, the phylo4d class. See also the checkPhylo4, the phylo4 constructor and the phylo4 class. See coerce-methods for translation functions.

geospiza Data from Darwin’s finches

Description

Phylogenetic tree and morphological data for Darwin’s finches, in different formats

Usage

data(geospiza)
data(geospiza_raw)

Format

ageospiza is a phylo4d object; geospiza_raw is a list containing tree, a phylo object (the tree), data, and a data frame with the data (for showing examples of how to merge tree and data)

Note

Stolen from Luke Harmon’s Geiger package, to avoid unnecessary dependencies

Source

Dolph Schluter via Luke Harmon

Examples

data(geospiza)
plot(geospiza)
getNode  

node and edge look-up functions

Description

Functions for retrieving node and edge IDs (possibly with corresponding labels) from a phylogenetic tree.

Usage

```r
getNode(x, node, type=c("all", "tip", "internal"),
       missing=c("warn", "OK", "fail"))
nodeId(x, type=c("all", "tip", "internal", "root"))
edgeId(x, type=c("all", "tip", "internal", "root"))
getEdge(x, node, type=c("descendant", "ancestor"),
        missing = c("warn", "OK", "fail"))
```

Arguments

- `x` a `phylo4` object (or one inheriting from `phylo4`, e.g. a `phylo4d` object)
- `node` either an integer vector corresponding to node ID numbers, or a character vector corresponding to node labels; if missing, all nodes appropriate to the specified type will be returned by `getNode`, and all edges appropriate to the specified type will be returned by `getEdge`.
- `type` (getNode) specify whether to return nodes matching "all" tree nodes (default), only "tip" nodes, or only "internal" nodes; (nodeId, edgeId) specify whether to return "all" tree nodes, or only those corresponding to "tip", "internal", or "root" nodes; (getEdge) specify whether to look up edges based on their descendant node ("descendant") or ancestral node ("ancestor")
- `missing` what to do if some requested node IDs or names are not in the tree: warn, do nothing, or stop with an error

Details

getNode and `getEdge` are primarily intended for looking up the IDs either of nodes themselves or of edges associated with those nodes. Note that they behave quite differently. With `getNode`, any input nodes are looked up against tree nodes of the specified type, and those that match are returned as numeric node IDs with node labels (if they exist) as element names. With `getEdge`, any input nodes are looked up against edge ends of the specified type, and those that match are returned as character edge IDs with the corresponding node ID as element names.

If `missing` is "warn" or "OK", `NA` is returned for any nodes that are unmatched for the specified type. This can provide a mechanism for filtering a set of nodes or edges.

`nodeId` provides similar output to `getNode` in the case when no node is supplied, but it is faster and returns an unnamed vector of the numeric IDs of all nodes of the specified node type. Similarly, `edgeId` simply returns an unnamed vector of the character IDs of all edges for which the descendant node is of the specified node type.
Value

getNode returns a named integer vector of node IDs, in the order of input nodes if provided, otherwise in nodeID order.

getEdge returns a named character vector of edge IDs, in the order of input nodes if provided, otherwise in nodeID order.

nodeID returns an unnamed integer vector of node IDs, in ascending order.

getEdge returns an unnamed character vector of edge IDs, in edge matrix order.

Examples

```r
data(geospiza)
nodelabels(geospiza) <- LETTERS[1:nNodes(geospiza)]
plot(as(geospiza, "phylo4"), show.node.label=TRUE)
getNode(geospiza, 18)
getNode(geospiza, "D")
getEdge(geospiza, "D")
getEdge(geospiza, "D", type="ancestor")

## match nodes only to tip nodes, flagging invalid cases as NA
getNode(geospiza, c(1, 18, 999), type="tip", missing="OK")

## get all edges that descend from internal nodes
getEdge(geospiza, type="ancestor")

## identify an edge from its terminal node
getEdge(geospiza, c("olivacea", "B", "fortis"))
getNode(geospiza, c("olivacea", "B", "fortis"))

geospiza$edge[c(26, 1, 11),]

## quickly get all tip node IDs and tip edge IDs
nodeID(geospiza, "tip")
edgeID(geospiza, "tip")
```

---

hasSingle

*Test trees for polytomies, inline nodes, or reticulation*

Description

Checks to see whether trees have (structural) polytomies, inline nodes (i.e., nodes with a single descendant), or reticulation (i.e., nodes with more than one ancestor).

Usage

```r
hasSingle(object)
hasPoly(object)
hasRetic(object)
```
Import Nexus and Newick files

Arguments

object an object inheriting from class phylo4

Value

Logical value

Note

Some algorithms are unhappy with structural polytomies (i.e., >2 descendants from a node), with single-descendant nodes, or with reticulation; these functions check those properties. We haven’t bothered to check for zero branch lengths: the consensus is that it doesn’t come up much, and that it’s simple enough to test any(edgeLength(x) == 0) in these cases. (Single-descendant nodes are used e.g. in OUCH, or in other cases to represent events occurring along a branch.)

Author(s)

Ben Bolker

Examples

```r
tree.owls.bis <- ape::read.tree(text = "((Strix_aluco:4.2,Asio_otos:4.2):3.1,Athene_noctua:7.3);")
owls4 <- as(tree.owls.bis, "phylo4")
hasPoly(nowls4)
hasSingle(nowls4)
```

Import Nexus and Newick files

Create a phylo4, phylo4d or data.frame object from a Nexus or a Newick file

Description

readNexus reads a Nexus file and outputs a phylo4 or phylo4d or data.frame object.
readNewick reads a Newick file and outputs a phylo4 or phylo4d object.

Usage

```r
readNexus(file, simplify=FALSE, type=c("all", "tree", "data"),
  char.all=FALSE, polymorphic.convert=TRUE, levels.uniform=FALSE,
  quiet=TRUE, check.node.labels=c("keep", "drop", "asdata"),
  return.labels=TRUE, check.names=TRUE, convert.edge.length=FALSE,
  ...)
```

```r
readNewick(file, simplify=FALSE, quiet=TRUE,
  check.node.labels=c("keep", "drop", "asdata"),
  convert.edge.length=FALSE, ...)
```
Import Nexus and Newick files

Arguments

**file**
a Nexus file for `readNexus` or a file that contains Newick formatted trees for `readNewick`

**simplify**
If there are multiple trees in the file, only the first one is returned if `TRUE` and a list of phylo4/phylo4d objects is returned if the file contains multiple trees.

**type**
Determines which type of objects to return, if present in the file (see Details).

**char.all**
If `TRUE`, returns all characters, even those excluded in the NEXUS file

**polymorphic.convert**
If `TRUE`, converts polymorphic characters to missing data

**levels.uniform**
If `TRUE`, uses the same levels for all characters

**quiet**
If `FALSE` the output of the NCL interface is printed. This is mainly for debugging purposes. This option can considerably slow down the process if the tree is big or there are many trees in the file.

**check.node.labels**
Determines how the node labels in the Nexus or Newick files should be treated in the phylo4 object, see Details for more information.

**return.labels**
Determines whether state names (if `TRUE`) or state codes should be returned.

**check.names**
logical. If `TRUE` then the names of the characters from the NEXUS file are checked to ensure that they are syntactically valid variable names and are not duplicated. If necessary they are adjusted (by `make.names`) so that they are.

**convert.edge.length**
logical. If `TRUE` negative edge lengths are replaced with 0. At this time `phylobase` does not accept objects with negative branch lengths, this workaround allows to still use trees with negative branch lengths as an artifact of the method used to build the tree.

... Additional arguments to be passed to phylo4 or phylo4d constructor (see Details)

Details

`readNexus` extracts data held in a Nexus file, specifically from DATA, CHARACTER or TREES blocks present in the file. The `type` argument specifies which of these is returned:

- **data** will only return a `data.frame` of the contents of all DATA and CHARACTER blocks.
- **tree** will only return a phylo4 object of the contents of the TREES block.
- **all** if only data or a tree are present in the file, this option will act as the options above, returning either a `data.frame` or a phylo4 object respectively. If both are present then a phylo4d object is returned containing both.

The function returns NULL if the type of data requested is not present in the file, or if neither data nor tree blocks are present.

Depending on the context `readNexus` will call either the phylo4 or phylo4d constructor. In addition with `type="all"`, the phylo4d constructor will be used if `check.node.labels="asdata"`.

`readNewick` imports newick formatted tree files and will return a phylo4 or a phylo4d object if the option `check.node.labels="asdata"` is invoked.

For both `readNexus` and `readNewick`, the options for `check.node.labels` can take the values:
multiPhylo-class

keep  the node labels of the trees will be passed as node labels in the phylo4 object
drop the node labels of the trees will be ignored in the phylo4 object
asdata the node labels will be passed as data and a phylo4d object will be returned.

If you use the option asdata on a file with no node labels, a warning message is issued, and thus
check.node.labels takes the value drop.
For both readNexus and readNewick, additional arguments can be passed to the constructors such as
annot, missing.data or extra.data. See the documentation of phylo4-methods, phylo4d and
formatData for the complete list of options.

Value

Depending on the value of type and the contents of the file, one of: a data.frame, a phylo4
object, a phylo4d object or NULL. If several trees are included in the Nexus file and the option
simplify=FALSE a list of phylo4 or phylo4d objects is returned.

Note

Underscores in state labels (i.e. trait or taxon names) will be translated to spaces when read by NCL.
Unless check.names=FALSE, trait names will be converted to valid R names (see make.names) on
input to R, so spaces will be translated to periods.

Note

This relies on Version 2.1.13 of the Nexus Class Library by Paul Lewis and Mark Holder. Tree
reading is done initially using internal APE functions; these functions can be confused if tree names
include the word (“tree”) in them.

Author(s)

Brian O’Meara, Francois Michonneau, Derrick Zwickl

See Also

the phylo4d class, the phylo4 class

multiPhylo-class  multiPhylo4 and extended classes

Description

Classes for lists of phylogenetic trees. These classes and methods are planned for a future version
of phylobase.
**Description**

Method to retrieve the number of datasets associated with a phylogenetic tree stored as a phylo4d object.

**Usage**

```r
## S4 method for signature 'phylo4d'
ndata(x)
```

**Arguments**

- `x`: A phylo4d object

**Details**

`ndata` returns the number of datasets (i.e., columns) that are associated with a phylo4d object.

**Value**

`ndata` returns a vector.

**Author(s)**

Francois Michonnea

**See Also**

tdata, phylo4d

**Examples**

```r
data(geospiza)
ndata(geospiza)
```
Description

A tiny tree, for testing/example purposes, using one of the examples from the ape package

Usage

data(owls4)

Format

This is the standard ‘owls’ tree from the ape package, in phylo4 format.

Source

From various examples in the ape package

Examples

data(owls4)
plot(owls4)

pdata

Constructor for pdata (phylogenetic data) class

Description

Combine data, type, comments, and metadata information to create a new pdata object, or check such an object for consistency

Usage

data(pdata(data, type, comment, metadata)
check_pdata(object)

Arguments

data a data frame
type a factor with levels as specified by pdata, the same length as ncol(data)
comment a character vector, the same length as ncol(data)
metadata an arbitrary list
object an object of class pdata
Value
An object of class pdata

Author(s)
Ben Bolker

See Also
pdata

---

**pdata-class**

**Class "pdata"**

**Description**
Data class for phylo4d objects

**Objects from the Class**
Objects can be created by calls of the form `new("pdata", ...).

**Slots**
- **data**: A data frame of tip or node data. Can be accessed transparently with any of the data frame accessor methods
- **type**: A factor with length equal to `nrow(data)` and levels("multitype", "binary", "continuous", "DNA", "RNA", "aacid")
- **comment**: A character vector of length `nrow(data)`
- **metadata**: An arbitrary list, for storing other user-defined metadata

**Methods**
- `[ signature(x = "pdata")`: access data rows, columns or elements
- `[<- signature(x = "pdata")`: set data rows, columns or elements
- `[<- signature(x = "pdata")`: set data columns or elements
- `[[ signature(x = "pdata", i = "ANY", j = "ANY")`: access data columns or elements
- `[[ signature(x = "pdata", i = "ANY", j = "missing")`: set data columns or elements

**Author(s)**
Ben Bolker
phylo4-accessors Methods for S4 phylogeny classes

Description

Generic methods for phylogenetic trees represented as S4 classes

Usage

```r
## S4 method for signature 'phylo4'
nNodes(x)
## S4 method for signature 'phylo4'
nTips(x)
## S4 method for signature 'phylo4'
depthTips(x)
## S4 method for signature 'phylo4'
depthTips(x)
## S4 method for signature 'phylo4'
edges(x, drop.root=FALSE, ...)
## S4 method for signature 'phylo4'
nEdges(x)
## S4 method for signature 'phylo4'
edgeOrder(x, ...)
## S4 method for signature 'phylo4'
hasEdgeLength(x)
## S4 method for signature 'phylo4'
edgeLength(x, node)
## S4 replacement method for signature 'phylo4'
edgeLength(x, use.names=TRUE) <- value
## S4 method for signature 'phylo4'
nodeType(x)
## S4 method for signature 'phylo4'
nodeDepth(x, node)
## S4 method for signature 'phylo4'
isRooted(x)
## S4 method for signature 'phylo4'
rootEdge(x)
## S4 method for signature 'phylo4'
rootNode(x)
## S4 replacement method for signature 'phylo4'
rootNode(x) <- value
## S4 method for signature 'phylo4'
isUltrametric(x, tol=Machine$double.eps^0.5)
```

Arguments

- `x` a phylo4/phylo4d object
node which edge to extract (indexed by descendant node)
value a vector of edge lengths or a node number
use.names Should the names of value be used to match edge lengths provided?
drop.root logical: drop root row from edge matrix?
tol tolerance in rounding error to determine whether the tree is ultrametric
... additional parameters passed (currently ignored)

Methods

data(geospiza)
dexelength(geospiza, 5)
dexelength(geospiza, "olivacea")
dexelength(geospiza, 5:7)
The phylo4 class

Description

Classes for phylogenetic trees

Objects from the Class

Phylogenetic tree objects can be created by calls to the phylo4 constructor function. Translation functions from other phylogenetic packages are also available. See coerce-methods.

Slots

edge: Matrix of edges
edge.label: Character vector of edge (branch) labels
edge.length: Numeric vector of edge (branch) lengths
label: Character vector of tip (and optionally internal) node labels
order: character: tree ordering (allowable values are listed in phylo4_orderings, currently "unknown", "preorder" (="cladewise" in ape), and "postorder", with "cladewise" and "pruning-wise" also allowed for compatibility with ape)
annotate: annotation data for tree (currently unstructured/unused by methods)
metadata: metadata for node/tip data (currently unstructured/unused by methods)

Author(s)

Ben Bolker, Thibaut Jombart

See Also

The phylo4 constructor, the checkPhylo4 function to check the validity of phylo4 objects. See also the phylo4d constructor and the phylo4d class.

Displaying phylo4 object

Description

Display methods for phylo4 and phylo4d phylogenetic trees
Usage

```r
## S4 method for signature 'phylo4'
print(x, edgeOrder = c("pretty", "real"), printall)
## S4 method for signature 'phylo4'
summary(object, quiet=FALSE)
## S4 method for signature 'phylo4'
show(object)
## S4 method for signature 'phylo4'
names(x)
```

Arguments

- `x` a phylo4 object
- `object` a phylo4 object
- `edgeOrder` Character string indicating whether the edges should be printed as ordered in the tree "real" (e.g. preorder or postorder), or "pretty" printed with tips collated together
- `printall` If TRUE all tip labels are printed
- `quiet` a logical stating whether the results of the summary should be printed to the screen (FALSE, default) or not (TRUE)

Value

The `summary` method invisibly returns a list with the following components:

- `name` the name of the object
- `nb.tips` the number of tips
- `nb.nodes` the number of nodes
- `mean.el` mean of edge lengths
- `var.el` variance of edge lengths (estimate for population)
- `sumry.el` summary (i.e. range and quartiles) of the edge lengths
- `degree` (optional) degree (i.e. number of descendants) of each node; displayed only when there are polytomies
- `polytomy` (optional) type of polytomy for each node: ‘node’, ‘terminal’ (all descendants are tips) or ‘internal’ (at least one descendant is an internal node); displayed only when there are polytomies

The `names` method returns a vector of characters corresponding to the names of the slots.

Methods

- `print` signature(x = "phylo4"): print method
- `show` signature(object = "phylo4"): show method
- `summary` signature(object = "phylo4"): summary method
- `names` signature(x = "phylo4"): gives the slot names
- `head` signature(object = "phylo4"): show first few nodes
- `tail` signature(object = "phylo4"): show last few nodes
phylo4-labels

Author(s)

Ben Bolker, Thibaut Jombart

See Also

The phylo4 constructor, the checkPhylo4 function to check the validity of phylo4 objects. See also the phylo4d constructor and the phylo4d class.

Examples

tree.owls <- ape::read.tree(text=tOwls)
P1 <- as(tree.owls, "phylo4")
P1
summary(P1)

### summary of a polytomous tree
E <- matrix(c(8, 9,
             9, 10,
             10, 1,
             10, 2,
             9, 3,
             9, 4,
             8, 11,
             11, 5,
             11, 6,
             11, 7,
             0, 8), ncol=2, byrow=TRUE)
P2 <- phylo4(E)
nodelabels(P2) <- as.character(nodeId(P2, "internal"))
plot(P2, show.node.label=TRUE)
sumryP2 <- summary(P2)
sumryP2

---

phylo4-labels Labels for phylo4/phylo4d objects

Description

Methods for creating, accessing and updating labels in phylo4/phylo4d objects
Usage

```r
## S4 method for signature 'phylo4'
labels(object, type=c("all", "tip", "internal"))
## S4 replacement method for signature 'phylo4,ANY,ANY,character'
labels(x, type=c("all", "tip", "internal"), use.names=FALSE) <- value

## S4 method for signature 'phylo4'
hasDuplicatedLabels(x, type=c("all", "tip", "internal"))

## S4 method for signature 'phylo4'
tipLabels(x)
## S4 replacement method for signature 'phylo4,character'
tipLabels(x) <- value

## S4 method for signature 'phylo4'
hasNodeLabels(x)
## S4 method for signature 'phylo4'
nodeLabels(x)
## S4 replacement method for signature 'phylo4,character'
nodeLabels(x) <- value

## S4 method for signature 'phylo4'
hasEdgeLabels(x)
## S4 method for signature 'phylo4'
edgeLabels(x)
## S4 replacement method for signature 'phylo4,character'
edgeLabels(x) <- value
```

Arguments

- `x`: a phylo4 or phylo4d object.
- `object`: a phylo4 or phylo4d object.
- `type`: which type of labels: all (tips and internal nodes), tip (tips only), internal (internal nodes only).
- `value`: a vector of class character, see Details for more information.
- `use.names`: should the names of the vector used to create/update labels be used to match the labels? See Details for more information.

Details

In phylo4/phylo4d objects, tips must have labels (that’s why there is no method for hasTipLabels), internal nodes and edges can have labels.

Labels must be provided as a vector of class character. The length of the vector must match the number of elements they label.
The option `use.names` allows the user to match a label to a particular node. In this case, the vector must have names that match the node numbers.

The function `labels` is mostly intended to be used internally.

**Methods**

- `labels` signature(object = "phylo4"): tip and/or internal node labels, ordered by node ID
- `hasDuplicatedLabels` signature(object = "phylo4"): are any labels duplicated?
- `tipLabels` signature(object = "phylo4"): tip labels, ordered by node ID
- `hasNodeLabels` signature(object = "phylo4"): whether tree has (internal) node labels
- `nodeLabels` signature(object = "phylo4"): internal node labels, ordered by node ID
- `hasEdgeLabels` signature(object = "phylo4"): whether tree has (internal) edge labels
- `edgeLabels` signature(object = "phylo4"): internal edge labels, ordered according to the edge matrix

**Examples**

```r
data(geospiza)

## Return labels from geospiza
tipLabels(geospiza)

## Internal node labels in geospiza are empty
nodeLabels(geospiza)

## Creating internal node labels
dlbl <- paste("n", 1:nNodes(geospiza), sep="")
nodeLabels(geospiza) <- dlbl
nodeLabels(geospiza)

## naming the labels
names(dlbl) <- nodeID(geospiza, "internal")

## shuffling the labels
(dlbl <- sample(dlbl))

## by default, the labels are attributed in the order they are given:
nodeLabels(geospiza) <- dlbl
nodeLabels(geospiza)

## but use.names puts them in the correct order
labels(geospiza, "internal", use.names=TRUE) <- dlbl
nodeLabels(geospiza)
```
Description

phylo4 is a generic constructor that creates a phylogenetic tree object for use in phylobase methods. Phylobase contains functions for input of phylogenetic trees and data, manipulation of these objects including pruning and subsetting, and plotting. The phylobase package also contains translation functions to forms used in other comparative phylogenetic method packages.

Usage

```r
## S4 method for signature 'matrix'
phylo4(x, edge.length = NULL, tip.label = NULL,
      node.label = NULL, edge.label = NULL, order="unknown",
      annotate=list())

## S4 method for signature 'phylo'
phylo4(x, check.node.labels = c("keep", "drop"),
      annotate=list())
```

Arguments

- `x`: a matrix of edges or an object of class phylo (see above)
- `edge`: A numeric, two-column matrix with as many rows as branches in the phylogeny.
- `edge.length`: Edge (branch) length. (Optional)
- `tip.label`: A character vector of species names (names of "tip" nodes). (Optional)
- `node.label`: A character vector of internal node names. (Optional)
- `edge.label`: A character vector of edge (branch) names. (Optional)
- `order`: character: tree ordering (allowable values are listed in phylo4_orderings, currently "unknown", "preorder" (="cladewise" in ape), and "postorder", with "clade-wise" and "pruningwise" also allowed for compatibility with ape)
- `check.node.labels`: if `x` is of class phylo, either "keep" (the default) or "drop" node labels. This argument is useful if the phylo object has non-unique node labels.
- `annotate`: any additional annotation data to be passed to the new object

Details

The minimum information necessary to create a phylobase tree object is a valid edge matrix. The edge matrix describes the topology of the phylogeny. Each row describes a branch of the phylogeny, with the (descendant) node number in column 2 and its ancestor’s node number in column 1. These numbers are used internally and must be unique for each node.

The labels designate either nodes or edges. The vector node.label names internal nodes, and together with tip.label, name all nodes in the tree. The vector edge.label names all branches
in the tree. All label vectors are optional, and if they are not given, internally-generated labels will be assigned. The labels, whether user-specified or internally generated, must be unique as they are used to join species data with phylogenetic trees.

Methods

\[ \text{x = "matrix"} \] creates a phylobase tree from a matrix of edges

\[ \text{x = "phylo"} \] creates a phylobase tree from an object of class phylo

Note

Translation functions are available from many valid tree formats. See coerce-methods.

Author(s)

phylobase team

See Also

coerce-methods for translation functions. The phylo4 class, the formatData function to check the validity of phylo4 objects. See also the phylo4d constructor, and phylo4d class.

Examples

```
# a three species tree:
mytree <- phylo4(x=matrix(data=c(4,1, 4,5, 5,2, 5,3, 0,4), ncol=2,
byrow=TRUE), tip.label=c("speciesA", "speciesB", "speciesC"))
mytree
plot(mytree)

# another way to specify the same tree:
mytree <- phylo4(x=cbind(c(4, 4, 5, 5, 0), c(1, 5, 2, 3, 4)),
tip.label=c("speciesA", "speciesB", "speciesC"))

# another way:
mytree <- phylo4(x=rbind(c(4, 1), c(4, 5), c(5, 2), c(5, 3), c(0, 4)),
tip.label=c("speciesA", "speciesB", "speciesC"))

# with branch lengths:
mytree <- phylo4(x=rbind(c(4, 1), c(4, 5), c(5, 2), c(5, 3), c(0, 4)),
tip.label=c("speciesA", "speciesB", "speciesC"), edge.length=c(1,.2,.8,.8,NA))
plot(mytree)
```
phylo4d

*Combine a phylogenetic tree with data*

**Description**

phylo4d is a generic constructor which merges a phylogenetic tree with data frames to create a combined object of class phylo4d.

**Usage**

```r
## S4 method for signature 'phylo'
phylo4d(x, tip.data = NULL, node.data = NULL,
       all.data = NULL, check.node.labels = c("keep", "drop", "asdata"),
       annotate = list(), metadata = list(), ...)

## S4 method for signature 'phylo4'
phylo4d(x, tip.data = NULL, node.data = NULL,
       all.data = NULL, merge.data = TRUE, metadata = list(), ...)

## S4 method for signature 'matrix'
phylo4d(x, tip.data = NULL, node.data = NULL,
       all.data = NULL, merge.data = TRUE, metadata = list(),
       edge.length = NULL, tip.label = NULL, node.label = NULL,
       edge.label = NULL, order = "unknown", annotate = list(), ...)
```

**Arguments**

- `x`: an object of class phylo, phylo or a matrix of edges (see above)
- `tip.data`: a data frame (or object to be coerced to one) containing only tip data (Optional)
- `node.data`: a data frame (or object to be coerced to one) containing only node data (Optional)
- `all.data`: a data frame (or object to be coerced to one) containing both tip and node data (Optional)
- `merge.data`: if both `tip.data` and `node.data` are provided, should columns with common names will be merged together (default TRUE) or not (FALSE)? See details.
- `metadata`: any additional metadata to be passed to the new object
- `edge.length`: Edge (branch) length. (Optional)
- `tip.label`: A character vector of species names (names of "tip" nodes). (Optional)
- `node.label`: A character vector of internal node names. (Optional)
- `edge.label`: A character vector of edge (branch) names. (Optional)
- `order`: character: tree ordering (allowable values are listed in phylo4_orderings, currently "unknown", "preorder" (="cladewise" in ape), and "postorder", with "cladewise" and "pruningwise" also allowed for compatibility with ape)
- `annotate`: any additional annotation data to be passed to the new object
check.node.labels

if x is of class phylo, use either “keep” (the default) to retain internal node labels, “drop” to drop them, or “asdata” to convert them to numeric tree data. This argument is useful if the phylo object has non-unique node labels or node labels with informative data (e.g., posterior probabilities).

... further arguments to be passed to formatData. Notably, these additional arguments control the behavior of the constructor in the case of missing/extra data and where to look for labels in the case of non-unique labels that cannot be stored as row names in a data frame.

Details

You can provide several data frames to define traits associated with tip and/or internal nodes. By default, data row names are used to link data to nodes in the tree, with any number-like names (e.g., “10”) matched against node ID numbers, and any non-number-like names (e.g., “n10”) matched against node labels. Alternative matching rules can be specified by passing additional arguments to formatData; these include positional matching, matching exclusively on node labels, and matching based on a column of data rather than on row names. See formatData for more information.

Matching rules will apply the same way to all supplied data frames. This means that you need to be consistent with the row names of your data frames. It is good practice to use tip and node labels (or node numbers) when you combine data with a tree.

If you provide both tip.data and node.data, the treatment of columns with common names will depend on the merge.data argument. If TRUE, columns with the same name in both data frames will be merged; when merging columns of different data types, coercion to a common type will follow standard R rules. If merge.data is FALSE, columns with common names will be preserved independently, with “.tip” and “.node” appended to the names. This argument has no effect if tip.data and node.data have no column names in common.

If you provide all.data along with either of tip.data and node.data, it must have distinct column names, otherwise an error will result. Additionally, although supplying columns with the same names within data frames is not illegal, automatic renaming for uniqueness may lead to surprising results, so this practice should be avoided.

Value

An object of class phylo4d.

Methods

x = "phylo" merges a tree of class phylo with a data.frame into a phylo4d object
x = "matrix" merges a matrix of tree edges similar to the edge slot of a phylo object (or to £edge of a phylo object) with a data.frame into a phylo4d object
x = "phylo" merges a tree of class phylo with a data.frame into a phylo4d object

Note

Checking on matches between the tree and the data will be done by the validity checker (label matches between data and tree tips, number of rows of data vs. number of nodes/tips/etc.)
Author(s)

Ben Bolker, Thibaut Jombart, Steve Kembel, Francois Michonneau, Jim Regetz

See Also

coerce-methods for translation functions. The phylo4d class, the formatData function to check the validity of phylo4d objects; phylo4 class and phylo4 constructor.

Examples

tree.owls.bis <- ape::read.tree(text = treeOwls)
try(phylo4d(as(tree.owls.bis, "phylo4"), data.frame(wing = 1:3), silent = TRUE))
obj <- phylo4d(as(tree.owls.bis, "phylo4"), data.frame(wing = 1:3), match.data = FALSE)
obj
print(obj)

###
data(geospiza_raw)
geoTree <- geospiza_raw$tree
goData <- geospiza_raw$data

## fix differences in tip names between the tree and the data
goData <- rbind(goData, array(, dim = c(1, ncol(goData)),
dimnames = list("olivacea", colnames(goData))))

### Example using a tree of class 'phylo'
exGeo1 <- phylo4d(geoTree, tip.data = goData)

### Example using a tree of class 'phylo4'
geoTree <- as(geoTree, "phylo4")

## some random node data
rNodeData <- data.frame(randomTrait = rnorm(nNodes(geoTree)),
row.names = nodeId(geoTree, "internal"))
exGeo2 <- phylo4d(geoTree, tip.data = goData, node.data = rNodeData)

### Example using 'merge.data'
data(geospiza)
trGeo <- extractTree(geospiza)
Dt <- data.frame(a=rnorm(nTips(trGeo)), row.names = nodeId(trGeo, "tip"))
Dt <- data.frame(a=rnorm(nNodes(trGeo)), row.names = nodeId(trGeo, "internal"))

(matchData1 <- phylo4d(trGeo, tip.data = Dt, node.data = Dt, merge.data = FALSE))
(matchData2 <- phylo4d(trGeo, tip.data = Dt, node.data = Dt, merge.data = TRUE))

### Example with 'all.data'
nodeLabels(geoTree) <- as.character(nodeId(geoTree, "internal"))
rAllData <- data.frame(randomTrait = rnorm(nTips(geoTree) + nNodes(geoTree)),
row.names = labels(geoTree, "all"))
### phylo4d-class

```r
exGeo5 <- phylo4d(geoTree, all.data = rAllData)

## Examples using 'rownamesAsLabels' and comparing with match.data=FALSE
tDt <- data.frame(x = letters[1:nTips(trGeo)],
  row.names = sample(nodeId(trGeo, "tip")))
tipLabels(trGeo) <- as.character(sample(1:nTips(trGeo)))
(exGeo6 <- phylo4d(trGeo, tip.data = tDt, rownamesAsLabels = TRUE))
(exGeo7 <- phylo4d(trGeo, tip.data = tDt, rownamesAsLabels = FALSE))
(exGeo8 <- phylo4d(trGeo, tip.data = tDt, match.data = FALSE))

## generate a tree and some data
set.seed(1)
p3 <- ape::rcoal(5)
dat <- data.frame(a = rnorm(5), b = rnorm(5), row.names = p3$tip.label)
dat.defaultnames <- dat
row.names(dat.defaultnames) <- NULL
dat.superset <- rbind(dat, rnorm(2))
dat.subset <- dat[-1, ]

## create a phylo4 object from a phylo object
p4 <- as(p3, "phylo4")

## create phylo4d objects with tip data
p4d <- phylo4d(p4, dat)
###checkData(p4d)
p4d.sorted <- phylo4d(p4, dat[5:1, ])
try(p4d.nonames <- phylo4d(p4, dat.defaultnames))
p4d.nonames <- phylo4d(p4, dat.defaultnames, match.data = FALSE)

## Not run:
p4d.subset <- phylo4d(p4, dat.subset)
p4d.subset <- phylo4d(p4, dat.subset)
try(p4d.superset <- phylo4d(p4, dat.superset))
p4d.superset <- phylo4d(p4, dat.superset)

## End(Not run)

## create phylo4d objects with node data
nod.dat <- data.frame(a = rnorm(4), b = rnorm(4))
p4d.nod <- phylo4d(p4, node.data = nod.dat, match.data = FALSE)

## create phylo4 objects with node and tip data
p4d.all1 <- phylo4d(p4, node.data = nod.dat, tip.data = dat, match.data = FALSE)
nodeLabels(p4) <- as.character(nodeId(p4, "internal"))
p4d.all2 <- phylo4d(p4, all.data = rbind(dat, nod.dat, match.data = FALSE))
```
Description

S4 class for phylogenetic tree and data.

Objects from the Class

Objects can be created from various trees and a data.frame using the constructor `phylo4d`, or using `new("phylo4d", ...)` for empty objects.

Slots

- `edge`: Matrix of edges
- `edge.label`: Character vector of edge (branch) labels
- `edge.length`: Numeric vector of edge (branch) lengths
- `label`: Character vector of tip (and optionally internal node) labels
- `data`: data frame for traits of tips and internal nodes
- `order`: character: tree ordering (allowable values are listed in `phylo4_orderings`, currently "unknown", "preorder" (="cladewise" in ape), and "postorder", with "cladewise" and "pruning-wise" also allowed for compatibility with ape)
- `annote`: annotation data for tree (currently unstructured/unused by methods)
- `metadata`: metadata for node/tip data (currently unstructured/unused by methods)

Author(s)

Ben Bolker, Thibaut Jombart

See Also

`coerce-methods` for translation functions. The `phylo4d` constructor and the `formatData` function to check the validity of trees and data. See also the `phylo4` constructor, the `phylo4` class, and the `checkPhylo4` function to check the validity of phylo4 trees.

Examples

```r
example(read.tree, "ape")
obj <- phylo4d(as(tree.owls.bis, "phylo4"), data.frame(wing=1:3))
obj
names(obj)
summary(obj)
```
Display methods for phylo4d objects

Description

Methods used to display information about the data and the tree for phylo4d objects.

Usage

```r
## S4 method for signature 'phylo4d'
print(x, edgeOrder = c("pretty", "real"),
      printall=TRUE)
## S4 method for signature 'phylo4d'
summary(object, quiet=FALSE)
## S4 method for signature 'phylo4d'
names(x)
```

Arguments

- `x` a phylo4d object
- `object` a phylo4d object
- `edgeOrder` Character string indicating whether the edges should be printed as ordered in the tree "real" (e.g. preorder or postorder), or "pretty" printed with tips collated together
- `quiet` Should the summary be displayed on screen?
- `printall` If TRUE all tip labels are printed

Value

The `summary` method invisibly returns a list with the following components:

- `name` the name of the object
- `nb.tips` the number of tips
- `nb.nodes` the number of nodes
- `mean.el` mean of edge lengths
- `var.el` variance of edge lengths (estimate for population)
- `sumry.el` summary (i.e. range and quartiles) of the edge lengths
- `degree` (optional) type of polytomy for each node: ‘node’, ‘terminal’ (all descendants are tips) or ‘internal’ (at least one descendant is an internal node); displayed only when there are polytomies
- `sumry.tips` (optional) summary for the data associated with the tips
- `sumry.nodes` (optional) summary for the data associated with the internal nodes

The `names` method returns a vector of characters corresponding to the names of the slots.
Methods

- **print** signature(x = "phylo4d"): print method
- **show** signature(object = "phylo4d"): show method
- **summary** signature(object = "phylo4d"): summary method
- **names** signature(x = "phylo4d"): gives the slots names
- **head** signature(object = "phylo4d"): show first few nodes
- **tail** signature(object = "phylo4d"): show last few nodes

Author(s)

Ben Bolker, Thibaut Jombart

See Also

phylo4d constructor and phylo4d class.

---

**phylo4d-hasData** Tests for presence of data associated with trees stored as phylo4d objects

### Description

Methods that test for the presence of data associated with trees stored as phylo4d objects.

### Usage

```r
## S4 method for signature 'phylo4d'
hasNodeData(x)
## S4 method for signature 'phylo4d'
hasTipData(x)
```

### Arguments

- **x** a phylo4d object

### Details

The outcome of the test is based on row names of the data frame stored in data. If there are no rows having row names from the set `nodeId(x, "tip")`, then `hasTipData` returns `FALSE`. Likewise, if there are no rows having row names from the set `nodeId(x, "internal")`, then `hasNodeData` returns `FALSE`.

### Value

- **logical** return `TRUE` or `FALSE` depending whether data are associated with the tree (i.e., the slots `tip.data` or `node.data` are not empty)
Methods

- **hasNodeData** signature(object = "phylo4d"): whether tree has internal node data
- **hasTipData** signature(object = "phylo4d"): whether tree has data associated with its tips

Author(s)

Ben Bolker, Thibault Jombart, Francois Michonneau

See Also

- phylo4d constructor and phylo4d class.

Examples

```r
data(geospiza)
hasTipData(geospiza) # TRUE
hasNodeData(geospiza) # FALSE
```

Description

Provides a mean to control the validity of phylobase objects such as singletons, reticulated trees, polytomies, etc.

Usage

```r
phylobase.options(...)
```

Arguments

... a list may be given as the only argument, or any number of arguments may be in the name=value form, or no argument at all may be given. See the Value and Details sections for explanation. Arguments which are set by a function call will remain in effect until the end of the current session, unless overwritten by a subsequent call. In addition, they can be added as optional parameters of calls to specific functions of phylobase; in this case, their effect is limited to that function call. See the documentation of specific functions for the list of options which are recognised by that function.

- retic"warn", "fail" or "OK". Are reticulated trees allowed? "warn" returns a warning (default) and "fail" returns an error message
- singleton"warn", "fail" or "OK". Are singleton nodes allowed? "warn" returns a warning (default) and "fail" returns an error message
- multiroot"warn", "fail" or "OK". Are multiple roots allowed? "warn" returns a warning (default) and "fail" returns an error message
• poly"warn", "fail" or "OK". Are polytomies allowed? “warn” returns a warning (default) and “fail” returns an error message
• allow.duplicated.labels“warn”, “fail” or “OK”. Are duplicated labels allowed? “warn” returns a warning and “fail” (default) returns an error message

Details
The parameter values set via a call to this function will remain in effect for the rest of the session, affecting the subsequent behavior of phylobase.

Value
A list with the updated values of the parameters. If arguments are provided, the returned list is invisible.

Author(s)
Francois Michonneau (adapted from the package sm)

Examples
## Not run:
phylobase.options(poly="fail")
# subsequent trees with polytomies will fail the validity check
## End(Not run)

---

**phylobubbles**  

*Bubble plots for phylo4d objects*

Description
Plots either circles or squares corresponding to the magnitude of each cell of a phylo4d object.

Usage

```r
phylobubbles(type, place.tip.label, show.node.label, rot,
edge.color, node.color, tip.color, edge.width, newpage, ..., XXYY,
square = FALSE, grid = TRUE)
```

Arguments

- **type**  
  the type of plot
- **place.tip.label**  
  A string indicating whether labels should be plotted to the right or to the left of the bubble plot

## Not run:
phylobase.options(poly="fail")
# subsequent trees with polytomies will fail the validity check
## End(Not run)
show.node.label  A logical indicating whether internal node labels should be plotted
rot  The number of degrees that the plot should be rotated
delta.color  A vector of colors for the tree edge segments
delta.color  A vector of colors for the coloring the nodes
tit.color  A vector of colors for the coloring the tip labels
edge.width  A vector of line widths for the tree edges
newpage  Logical to control whether the device is cleared before plotting, useful for adding
plot inside other plots
...  Additional parameters passed to the bubble plotting functions
XXYY  The output from the phyloXXYY function
square  Logical indicating whether the plot 'bubbles' should be squares
gird  A logical indicating whether a grey grid should be plotted behind the bubbles

Author(s)
Peter Cowan <pdc@berkeley.edu>

See Also

 phyloXXYY, treePlot

Examples

### Should be DIRECTLY executable !! ----
### ==> Define data, use random,
### or do help(data-index) for the standard data sets.

| phylog-class | Class "phylog" |

Description

S4 version of the class phylog from the ade4 package.

Objects from the Class

A virtual Class: No objects may be created from it.

Extends

Class phylog, directly.
Methods

coerce signature(from = "phylo4", to = "phylog"): a phylo4 object can be coerced to a phylog object using as(object,"phylog").

Warning

phylog objects from ade4 are now deprecated; users are encouraged to use the adephylo package, which uses phylo and phylo4 objects directly.

Author(s)

Thibaut Jombart <jombart@biomserv.univ-lyon1.fr>

See Also

The original phylog from the ade4 package.

---

phylomat-class matrix classes for phylobase

Description

Classes representing phylogenies as matrices

Usage

as_phylo4vcov(from, ...)

Arguments

from a phylo4 object

... optional arguments, to be passed to vcov.phylo in ape (the main useful option is cor, which can be set to TRUE to compute a correlation rather than a variance-covariance matrix)

Objects from the Class

These are square matrices (with rows and columns corresponding to tips, and internal nodes implicit) with different meanings depending on the type (variance-covariance matrix, distance matrix, etc.).

Slots

.Data: square, numeric matrix with row and column labels corresponding to the tip labels

edge.label: character vector of edge labels

order: character describing original ordering of edge matrix
Author(s)
Ben Bolker

Examples

```r

tree.owls <- ape::read.tree(text=trString)
o2 <- as(tree.owls,"phylo4")
ov <- as(o2,"phylo4cov")
o3 <- as(ov,"phylo4")

## these are not completely identical, but are
## topologically identical ...

## edge matrices are in a different order:
## cf. o2@edge and o3@edge
## BUT the edge matrices are otherwise identical
identical(o2@edge[order(o2@edge[,2]),],
         o3@edge[order(o3@edge[,2]),])

## There is left/right ambiguity here in the tree orders:
## in o2 the 5->6->7->1 lineage
## (terminating in Strix aluco)
## is first, in o3 the 5->6->3 lineage
## (terminating in Athene noctua) is first.
```

Description

Calculates the node x and y locations for plotting a phylogenetic tree.

Usage

```r
phyloXXYY(phy, tip.order = NULL)
```

Arguments

- **phy** A `phylo4` or `phylo4d` object.
- **tip.order** A character vector of tip labels, indicating their order along the y axis (from top to bottom). Or, a numeric vector of tip node IDs indicating the order.

Details

The y coordinates of the tips are evenly spaced from 0 to 1 in pruningwise order. Ancestor y nodes are given the mean value of immediate descendants. The root is given the x coordinate 0 and descendant nodes are placed according to the cumulative branch length from the root, with a maximum x value of 1.
plotOneTree

Value

- `yy`: Internal node and tip y coordinates
- `xx`: Internal node and tip x coordinates
- `phy`: A phylo4 or phylo4d object
- `segs`: A list of `h0x`, `h1x`, `v0x`, `v1x` and `h0y`, `h1y`, `v0y`, `v1y` describing the start and end points for the plot line segments
- `torder`: The tip order provided as `tip.order` or if NULL the preoder tip order
- `eorder`: The an index of the reordered edges compared to the result of `edges(phy)`

Author(s)

Peter Cowan <pdc@berkeley.edu>

See Also

treePlot, plotOneTree

Examples

data(geospiza)
coor <- phyloXXXY(geospiza)
plot(coor$xx, coor$yy, pch = 20)

Description

Plots the phylogenetic tree contained in a phylo4 or phylo4d object.

Usage

plotOneTree(xxyy, type, show.tip.label, show.node.label, 
edge.color, node.color, tip.color, edge.width, rot)

Arguments

- `xxyy`: A list created by the `phyloXXXY` function
- `type`: A character string indicating the shape of plotted tree
- `show.tip.label`: Logical, indicating whether tip labels should be shown
- `show.node.label`: Logical, indicating whether node labels should be shown
- `edge.color`: A vector of colors in the order of `edges(phy)`
node.color A vector of colors indicating the colors of the node labels
tip.color A vector of colors indicating the colors of the tip labels
tip.color A vector in the order of edges(phy) indicating the widths of edge lines
rot Numeric indicating the rotation of the plot in degrees

Value

Returns no values, function invoked for the plotting side effect.

Author(s)

Peter Cowan <pdc@berkeley.edu>

See Also

treePlot, phyloXXYY

Examples

data(geospiza)
grid.newpage()
xxyy <- phyloXXYY(geospiza)
plotOneTree(xxyy, type = 'phylogram',
    show.tip.label = TRUE, show.node.label = TRUE,
    edge.color = 'black', node.color = 'orange', tip.color = 'blue',
    edge.width = 1, rot = 0
)

grid.newpage()
pushViewport(viewport(w = 0.8, h = 0.8))
plotOneTree(xxyy, type = 'phylogram',
    show.tip.label = TRUE, show.node.label = TRUE,
    edge.color = 'black', node.color = 'orange', tip.color = 'blue',
    edge.width = 1, rot = 0
)
popViewport()
**Arguments**

- `x` | a phylo4 tree or phylo4d tree+data object
- `edgeOrder` | in the data frame returned, the option 'pretty' returns the internal nodes followed by the tips, the option 'real' returns the nodes in the order they are stored in the edge matrix.
- `printall` | default prints entire tree. printall=FALSE returns the first 6 rows

**Details**

This is a user-friendly version of the tree representation, useful for checking that objects were read in completely and translated correctly. The phylogenetic tree is represented as a list of numbered nodes, linked in a particular way through time (or rates of evolutionary change). The topology is given by the pattern of links from each node to its ancestor. Also given are the taxon names, node type (root/internal/tip) and phenotypic data (if any) associated with the node, and the branch length from the node to its ancestor. A list of nodes (descendants) and ancestors is minimally required for a phylo4 object.

**Value**

A data.frame with a row for each node (descendant), sorted as follows: root first, then other internal nodes, and finally tips.

The returned data.frame has the following columns:

- `label` | Label for the taxon at the node (usually species name).
- `node` | Node number, i.e. the number identifying the node in x@edge.
- `ancestor` | Node number of the node’s ancestor.
- `branch.length` | The branch length connecting the node to its ancestor (NAs if missing).
- `node.type` | "root", "internal", or "tip". (internally generated)
- `data` | phenotypic data associated with the nodes, with separate columns for each variable.

**Note**

This is the default show() method for phylo4, phylo4d. It prints the user-supplied information for building a phylo4 object. For a full description of the phylo4 S4 object and slots, see `phylo4`.

**Author(s)**

Marguerite Butler Thibaut Jombart <jombart@biomserv.univ-lyon1.fr> Steve Kembel

**Examples**

```r
tree.phylo <- ape::read.tree(text="((a,b),c);")
tree <- as(tree.phylo, "phylo4")
#plot(tree,show.node=TRUE) ## plotting broken with empty node labels: FIXME
tip.data <- data.frame(size=c(1,2,3), row.names=c("a", "b", "c"))
```
reorder-methods

Description
Methods for reordering trees into various traversal orders

Usage
```r
## S4 method for signature 'phylo'
reorder(x, order = "cladewise")
## S4 method for signature 'phylo4'
reorder(x, order = c("preorder", "postorder"))
```

Arguments
- `x`: A phylo4 or phylo4d object
- `order`: The desired traversal order; currently only 'preorder' and 'postorder' are allowed for phylo4 and phylo4d objects, whereas only 'cladewise' and 'pruning-wise' are allowed for phylo objects

Details
The `reorder` method takes a phylo4 or phylo4d tree and orders the edge matrix (i.e. `edges(x)`) in the requested traversal order. Currently only two orderings are permitted, and both require rooted trees. In "postorder", a node’s descendants come before that node, thus the root, which is ancestral to all nodes, comes last. In "preorder", a node is visited before its descendants, thus the root comes first.

A method is also defined that takes an ape `phylo` object. This also takes an order argument, however, 'pruningwise' and 'cladewise' are the only acceptable parameters. This is because this method actually uses the ape `reorder()` command to complete the ordering.

Value
A phylo4 or phylo4d object with the edge, label, length and data slots ordered as `order`, which is itself recorded in the order slot.

Methods
- `x = "phylo"` reorders a phylo object
- `x = "phylo4"` reorders a phylo4 object
- `x = "phylo4d"` reorders a phylo4d object
Note

The "preorder" parameter corresponds to "cladewise" in the ape package, and "postorder" corresponds (almost but close enough?) to "pruningwise".


Author(s)

Peter Cowan, Jim Regetz

See Also

* reorder.phylo * in the ape package. ancestors ancestor siblings children descendants

Examples

```r
phy <- phylo4(ape::rtree(5))
edges(reorder(phy, "preorder"))
edges(reorder(phy, "postorder"))
```

Description

Methods for creating subsets of phylogenies, based on pruning a tree to include or exclude a set of terminal taxa, to include all descendants of the MRCA of multiple taxa, or to return a subtree rooted at a given node.

Usage

```r
## S4 method for signature 'phylo4'
subset(x, tips.include=NULL, tips.exclude=NULL,
       mrca=NULL, node.subtree=NULL, ...)
## S4 method for signature 'phylo4d'
subset(x, tips.include=NULL, tips.exclude=NULL,
       mrca=NULL, node.subtree=NULL, ...)

## S4 method for signature 'phylo4'
prune(x, tips.exclude, trim.internal = TRUE)
## S4 method for signature 'phylo4d'
prune(x, tips.exclude, trim.internal = TRUE)

## S4 method for signature 'phylo4'
x[i]
## S4 method for signature 'phylo4d'
x[i, j]
```
subset-methods

Arguments

x an object of class "phylo4" or "phylo4d"
tips.include A vector of tips to include in the subset tree
tips.exclude A vector of tips to exclude from the subset tree
mrca A vector of nodes for determining the most recent common ancestor, which is then used as the root of the subset tree
node.subtree A single internal node specifying the root of the subset tree
trim.internal A logical specifying whether to remove internal nodes that no longer have tip descendants in the subset tree
i ([] method) An index vector indicating tips to include
j ([] method, phylo4d only) An index vector indicating columns of node/tip data to include
... additional arguments to be passed to other methods

Details

The subset methods must be called using no more than one of the four main subsetting criteria arguments (tips.include, tips.exclude, mrca, or node.subtree). Each of these arguments can be either character or numeric. In the first case, they are treated as node labels; in the second case, they are treated as node numbers. For the first two arguments, any supplied tips not found in the tree (tipLabels(x)) will be ignored, with a warning. Similarly, for the mrca argument, any supplied tips or internal nodes not found in the tree will be ignored, with a warning. For the node.subtree argument, failure to provide a single, valid internal node will result in an error.

Although prune is mainly intended as the workhorse function called by subset, it may also be called directly. In general it should be equivalent to the tips.exclude form of subset (although perhaps with less up-front error checking).

The "[]" operator, when used as x[i], is similar to the tips.include form of subset. However, the indices used with this operator can also be logical, in which case the corresponding tips are assumed to be ordered as in node1d(x, "tip"), and recycling rules will apply (just like with a vector or a matrix). With a phylo4d object 'x', x[i,j] creates a subset of x taking i for a tip index and j for the index of data variables in tdata(geospiza, "all"). Note that the second index is optional: x[i, TRUE], x[i,], and x[i] are all equivalent.

Regardless of which approach to subsetting is used, the argument values must be such that at least two tips are retained.

If the most recent common ancestor of the retained tips is not the original root node, then the root node of the subset tree will be a descendant of the original root. For rooted trees with non-NA root edge length, this has implications for the new root edge length. In particular, the new length will be the summed edge length from the new root node back to the original root (including the original root edge). As an alternative, see the examples for a way to determine the length of the edge that was immediately ancestral to the new root node in the original tree.

Note that the correspondence between nodes and labels (and data in the case of phylo4d) will be retained after all forms of subsetting. Beware, however, that the node numbers (IDs) will likely be altered to reflect the new tree topology, and therefore cannot be compared directly between the original tree and the subset tree.
Value

an object of class "phylo4" or "phylo4d"

Methods

x = "phylo4"  subset tree

x = "phylo4d"  subset tree and corresponding node and tip data

Author(s)

Jim Regetz <regetz@nceas.ucsb.edu>
Steven Kembel <skembel@berkeley.edu>
Damien de Vienne <damien.de-vienne@u-psud.fr>
Thibaut Jombart <jombart@biomserv.univ-lyon1.fr>

Examples

data(geospiza)
nodelabels(geospiza) <- paste("N", nodeId(geospiza, "internal"), sep="")
geotree <- extractTree(geospiza)

## "subset" examples
tips <- c("difficilis", "fortis", "fuliginosa", "fusca", "olivacea",
"pallida", "parvulus", "scandens")
plot(subset(geotree, tips.include=tips))
plot(subset(geotree, tips.include=tips, trim.internal=FALSE))
plot(subset(geotree, tips.exclude="scandens"))
plot(subset(geotree, mrca=c("scandens","fortis","pauper")))
plot(subset(geotree, node.subtree=18))

## "prune" examples (equivalent to subset using tips.exclude)
plot(prune(geotree, tips))

## "[" examples (equivalent to subset using tips.include)
plot(geotree[c(1:6,14)])
plot(geospiza[c(1:6,14)])

## for phylo4d, subset both tips and data columns
geospiza[c(1:6,14), c("wingL", "beakD")]

## note handling of root edge length:
edgeLength(geotree)['0-15'] <- 0.1
geotree2 <- geotree[1:2]

## in subset tree, edge of new root extends back to the original root
edgeLength(geotree2)['0-3']

## edge length immediately ancestral to this node in the original tree
edgeLength(geotree, MRCA(geotree, tipLabels(geotree2)))
**Retrieving or updating tip and node data in phylo4d objects**

**Description**

Methods to retrieve or update tip, node or all data associated with a phylogenetic tree stored as a phylo4d object.

**Usage**

```r
## S4 method for signature 'phylo4d'
tdata(x, type=c("all", "tip", "internal"),
     label.type=c("row.names", "column"), empty.columns=TRUE)
## S4 replacement method for signature 'phylo4d,ANY'
tdata(x, type=c("all", "tip", "internal"),
      merge.data=TRUE, clear.all=FALSE, ...) <- value
## S4 method for signature 'phylo4d'
tipData(x, ...)
## S4 replacement method for signature 'phylo4d,ANY'
tipData(x, ...) <- value
## S4 method for signature 'phylo4d'
nodeData(x, ...)
## S4 replacement method for signature 'phylo4d,ANY'
nodeData(x, ...) <- value
```

**Arguments**

- **x** A phylo4d object
- **type** The type of data to retrieve or update: “all” (default) for data associated with both tip and internal nodes, “tip” for data associated with tips only, “internal” for data associated with internal nodes only.
- **label.type** How should the tip/node labels from the tree be returned? “row.names” returns them as row names of the data frame, “column” returns them in the first column of the data frame. This option is useful in the case of missing (NA) or non-unique labels.
- **empty.columns** Should columns filled with NA be returned?
- **merge.data** if tip or internal node data are provided and data already exists for the other type, this determines whether columns with common names will be merged together (default TRUE). If FALSE, columns with common names will be preserved separately, with “.tip” and “.node” appended to the names. This argument has no effect if tip and node data have no column names in common, or if type=“all”.
- **clear.all** If only tip or internal node data are to be replaced, should data of the other type be dropped?
- **...** For the tipData and nodeData accessors, further arguments to be used by tdata. For the replacement forms, further arguments to be used by formatData (e.g. match.data), see formatData for more details.
value a data frame (or object to be coerced to one) to replace the values associated with the nodes specified by the argument type

Value
tdata returns a data frame

Methods
tdata signature(object="phylo4d"): retrieve or update data associated with a tree in a phylo4d object

Author(s)
Ben Bolker, Thibaut Jombart, Francois Michonneau

See Also
phylo4d

Examples
data(geospiza)
tdata(geospiza)
tipData(geospiza) <- 1:nTips(geospiza)
tdata(geospiza)

Description
Plotting phylogenetic trees and associated data

Usage
tip.data.plot(xxyy, type = c("phylogram", "cladogram", "fan"), show.tip.label = TRUE, show.node.label = FALSE, rot = 0, tip.plot.fun = grid.points, edge.color = "black", node.color = "black", tip.color = "black", edge.width = 1, ...)
Arguments

- **xxyy**
  - A list created by the `phyloXXX` function

- **type**
  - A character string indicating the shape of plotted tree

- **show.tip.label**
  - Logical, indicating whether tip labels should be shown

- **show.node.label**
  - Logical, indicating whether node labels should be shown

- **rot**
  - Numeric indicating the rotation of the plot in degrees

- **tip.plot.fun**
  - A function used to plot the data elements of a `phylo4d` object

- **edge.color**
  - A vector of colors in the order of `edges(phy)`

- **node.color**
  - A vector of colors indicating the colors of the node labels

- **tip.color**
  - A vector of colors indicating the colors of the tip labels

- **edge.width**
  - A vector in the order of `edges(phy)` indicating the widths of edge lines

- **...**
  - Additional parameters passed to `tip.plot.fun`

Value

creates a plot on the current graphics device.

Author(s)

Peter Cowan

Description

Plot `phylo4` or `phylo4d` objects, including associated data.

Usage

```r
## S4 method for signature 'phylo4,phylo4d'
treePlot(phy, type = c("phylogram", " cladogram", "fan"), show.tip.label = TRUE,
        show.node.label = FALSE, tip.order = NULL, plot.data = is(phy, "phylo4d"),
        rot = 0, tip.plot.fun = "bubbles", edge.color = "black",
        node.color = "black", tip.color = "black", edge.width = 1, newpage = TRUE, ...)
```
Arguments

- **phy**: A phylo4 or phylo4d object
- **type**: A character string indicating the shape of plotted tree
- **show.tip.label**: Logical, indicating whether tip labels should be shown
- **show.node.label**: Logical, indicating whether node labels should be shown
- **tip.order**: If NULL, the tree is plotted with tips in preorder, if "rev" this is reversed. Otherwise, it is a character vector of tip labels, indicating their order along the y axis (from top to bottom). Or, a numeric vector of tip node IDs indicating the order.
- **plot.data**: Logical indicating whether phylo4d data should be plotted
- **rot**: Numeric indicating the rotation of the plot in degrees
- **tip.plot.fun**: A function used to generate plot at each tip of the phylogenetic trees
- **edge.color**: A vector of colors in the order of edges(phy)
- **node.color**: A vector of colors indicating the colors of the node labels
- **tip.color**: A vector of colors indicating the colors of the tip labels
- **edge.width**: A vector in the order of edges(phy) indicating the widths of edge lines
- **newpage**: Logical indicating whether the page should be cleared before plotting

Value

No return value, function invoked for plotting side effect

Methods

- **phy = "phylo4"** plots a tree of class phylo4
- **phy = "phylo4d"** plots a tree with one or more quantitative traits contained in a phylo4d object.

Author(s)

Peter Cowan <pdc@berkeley.edu>

See Also

- phylobubbles

Examples

```r
## example of plotting two grid plots on the same page
data(geospiza)
geotree <- extractTree(geospiza)
grid.newpage()
pushViewport(viewport(layout=grid.layout(nrow=1, ncol=2, name="base")))
pushViewport(viewport(layout.pos.col=1, name="plot1"))
```

tree traversal and utility functions

Description

Functions for describing relationships among phylogenetic nodes (i.e. internal nodes or tips).

Usage

ancestors(phy, node, type=c("all","parent","ALL"))
ancestor(phy, node)
siblings(phy, node, include.self=FALSE)
children(phy, node)
descendants(phy, node, type=c("tips","children","all"))
MRCA(phy, ...) 
shortestPath(phy, node1, node2)
## S4 method for signature 'phylo4'
sumEdgeLength(x, node)

Arguments

phy  
a phylo4 object (or one inheriting from phylo4, e.g. a phylo4d object)

x  
a phylo4 object (or one inheriting from phylo4, e.g. a phylo4d object)

node  
either an integer corresponding to a node ID number, or a character corresponding to a node label; for ancestors and descendants, this may be a vector of multiple node numbers or names

type  
(ancestors) specify whether to return just direct ancestor ("parent"), all ancestor nodes ("all"), or all ancestor nodes including self ("ALL"); (descendants) specify whether to return just direct descendants ("children"), all extant descendants ("tips"), or all descendant nodes ("all")

include.self  
whether to include self in list of siblings

...  
a list of node numbers or names, or a vector of node numbers or names

node1  
a node number (or name)

node2  
a node number (or name)
Details

ancestors and descendants can take node vectors of arbitrary length, returning a list of output vectors if the number of valid input nodes is greater than one. List element names are taken directly from the input node vector.

If any supplied nodes are not found in the tree, the behavior currently varies across functions. Invalid nodes are automatically omitted by ancestors and descendants, with a warning. ancestor will return NA for any invalid nodes, with a warning. Both children and siblings will return an empty vector, again with a warning. In contrast, MRCA and shortestPath will throw an immediate error if any input nodes are invalid.

Value

ancestors and descendants
return a named vector (or a list of such vectors in the case of multiple input nodes) of the ancestors and descendants of a node

ancestor and children
ancestor is analogous to ancestors(..., type="parent") (i.e. direct ancestor only), but returns a single concatenated vector in the case of multiple input nodes; children is analogous to descendants(..., type="children") (i.e. direct descendants only), but is not currently intended to be used with multiple input nodes

siblings
returns sibling nodes (children of the same parent)

MRCA
returns the most recent common ancestor of two or more nodes

shortestPath
returns the nodes of the shortest path from one node to another (excluding node1 and node2)

sumEdgeLength
returns the sum of branch length for branches starting at nodes provided

Note

MRCA is uppercase to avoid conflict with mrca in ape

See Also

mrca, in the ape package, gives a list of all subtrees

Examples

data(geospiza)
nodeLabels(geospiza) <- LETTERS[1:nNodes(geospiza)]
plot(as(geospiza, "phylo4"), show.node.label=TRUE)
ancestor(geospiza, "E")
children(geospiza, "C")
descendants(geospiza, "D", type="tips")
descendants(geospiza, "D", type="all")
ancestors(geospiza, "D")
MRCA(geospiza, "conirostris", "difficilis", "fuliginosa")
MRCA(geospiza, "olivacea", "conirostris")
## shortest path between 2 nodes
```
shortestPath(geospiza, "fortis", "fuliginosa")
shortestPath(geospiza, "F", "L")
```

## branch length from a tip to the root
```
sumEdgeLength(geospiza, ancestors(geospiza, "fortis", type="ALL"))
```
Index

*Topic **classes**
  - multiPhylo-class, 15
  - pdata-class, 18
  - phylo4-class, 21
  - phylo4-methods, 26
  - phylo4-class, 32
  - phylog-class, 37
  - phylomat-class, 38

*Topic **datasets**
  - geospiza, 10
  - owlsT, 17

*Topic **manip**
  - subset-methods, 44

*Topic **methods**
  - addData, 3
  - as, 5
  - extractTree, 8
  - nData, 16
  - phylo4-accessors, 19
  - phylo4-display, 21
  - phylo4-display, 33
  - phylo4-hasData, 34
  - phylodbubbles, 36
  - phyloXXYY, 39
  - plotOneTree, 40
  - printphylo4, 41
  - reorder-methods, 43
  - subset-methods, 44
  - tdata, 47
  - tip.data.plot, 48
  - treePlot-methods, 49

*Topic **misc**
  - checkPhylo4, 6
  - formatData, 9
  - getNode, 11
  - hasSingle, 12
  - Import Nexus and Newick files, 13
  - pdata, 17
  - phylo4d, 28
  - treewalk, 51

*Topic **package**
  - phylobase-package, 3

*Topic **phylobase**
  - phylobase.options, 35

*Topic **validator**
  - phylobase.options, 35
  - [,pdata,ANY,ANY,ANY-method (pdata-class), 18
  - [,pdata-method (pdata-class), 18
  - [,phylo4,ANY,ANY-method (subset-methods), 44
  - [,phylo4,character,missing,missing-method (subset-methods), 44
  - [,phylo4,logical,missing,missing-method (subset-methods), 44
  - [,phylo4,missing,missing,missing-method (subset-methods), 44
  - [,phylo4,numeric,missing,missing-method (subset-methods), 44
  - [,phylo4-method (subset-methods), 44
  - [,phylo4d,ANY,character,missing-method (subset-methods), 44
  - [,phylo4d,logical,missing-method (subset-methods), 44
  - [,phylo4d,missing,missing-method (subset-methods), 44
  - [,phylo4d-method (subset-methods), 44
  - [,phylo4d-method (subset-methods), 44
  - [-methods (subset-methods), 44
  - [-,pdata-method (pdata-class), 18
  - [,pdata,ANY,ANY-method (pdata-class), 18
  - [,pdata-method (pdata-class), 18
  - [<-,pdata-method (pdata-class), 18
  - <+plyo4d-method (pdata-class), 18
  - <+plyo4d-method (pdata-class), 18
  - addData, 3
INDEX

hasEdgeLabels (phylo4-labels), 23
hasEdgeLabels, phylo4-method
(phylo4-labels), 23
hasEdgeLabels-methods (phylo4-labels), 23
hasEdgeLength (phylo4-accessors), 19
hasEdgeLength, phylo4-method
(phylo4-accessors), 19
hasEdgeLength-methods
(phylo4-accessors), 19
hasNodeData (phylo4d-hasData), 34
hasNodeData, phylo4d-method
(phylo4d-hasData), 34
hasNodeData-methods (phylo4d-hasData), 34
hasNodeLabels (phylo4-labels), 23
hasNodeLabels, phylo4-method
(phylo4-labels), 23
hasNodeLabels-methods (phylo4-labels), 23
hasPoly (hasSingle), 12
hasRetic (hasSingle), 12
hasSingle, 12
hasTipData (phylo4d-hasData), 34
hasTipData, phylo4d-method
(phylo4d-hasData), 34
hasTipData-methods (phylo4d-hasData), 34
head, phylo4-method (phylo4-display), 21
head, phylo4d-method (phylo4d-display), 33

Import Nexus and Newick files, 13
isRooted (phylo4-accessors), 19
isRooted, phylo4-method
(phylo4-accessors), 19
isRooted-methods (phylo4-accessors), 19
isUltrametric (phylo4-accessors), 19
isUltrametric, phylo4-method
(phylo4-accessors), 19
isUltrametric-methods
(phylo4-accessors), 19

labels, phylo4-method (phylo4-labels), 23
labels< (phylo4-labels), 23
labels<, phylo4, ANY, ANY, character-method
(phylo4-labels), 23
labels<, phylo4d, ANY, ANY, character-method
(phylo4-labels), 23

make.names, 15
MRCA (treewalk), 51
mrca, 52
multiPhylo-class, 15
multiPhylo4, 5
multiPhylo4-class (multiPhylo-class), 15
multiPhylo4d-class (multiPhylo-class), 15

na.omit, phylo4d-method
(subset-methods), 44
names, phylo4-method (phylo4-display), 21
names, phylo4d-method (phylo4d-display), 33

nData, 16
nData, phylo4d-method (nData), 16
nEdges (phylo4-accessors), 19
nEdges, phylo4-method
(phylo4-accessors), 19
nEdges-methods (phylo4-accessors), 19
nNodes (phylo4-accessors), 19
nNodes, phylo4-method
(phylo4-accessors), 19
nNodes-methods (phylo4-accessors), 19
nodeData (tdata), 47
nodeData, phylo4d-method (tdata), 47
nodeData-method (tdata), 47
nodeData< (tdata), 47
nodeData<, phylo4d, ANY-method (tdata), 47

nodeData<, phylo4d-method (tdata), 47
nodeDepth (phylo4-accessors), 19
nodeDepth, phylo4-method
(phylo4-accessors), 19
nodeDepth-methods (phylo4-accessors), 19
nodeId (getNode), 11
nodeId, phylo4-method (getNode), 11
nodeLabels (phylo4-labels), 23
nodeLabels, phylo4-method
(phylo4-labels), 23
nodeLabels-methods (phylo4-labels), 23
nodeLabels< (phylo4-labels), 23
nodeLabels<, phylo4, character-method
(phylo4-labels), 23
nodeLabels<, phylo4d, ANY-method
(phylo4-labels), 23

nodeType (phylo4-accessors), 19
nodeType, phylo4-method
(phylo4-accessors), 19
nTips (phylo4-accessors), 19
nTips, phylo-method (phylo4-accessors), 19
nTips, phylo4-method (phylo4-accessors), 19
nTips-methods (phylo4-accessors), 19
owls4, 17
pdata, 17, 17, 18
pdata-class, 18
phylo-class (phylo4-class), 21
phylo4, 5-8, 10, 11, 15, 21, 23, 27, 30, 32, 38, 42, 43, 50, 51
phylo4 (phylo4-methods), 26
phylo4, matrix-method (phylo4-methods), 26
phylo4, phylo-method (phylo4-methods), 26
phylo4-accessors, 19
phylo4-class, 21
phylo4-display, 21
phylo4-labels, 23
phylo4-methods, 15, 26
phylo4-orderings (phylo4-class), 21
phylo4d, 4-8, 10, 11, 15, 16, 21, 23, 27, 28, 29, 30, 32, 34, 35, 43, 45, 48, 50, 51
phylo4d, matrix-method (phylo4d), 28
phylo4d, phylo-method (phylo4d), 28
phylo4d, phylo4d-method (phylo4d), 28
phylo4d-class, 31
phylo4d-display, 33
phylo4d-hasData, 34
phylo4d-methods (phylo4d), 28
phylo4vcov-class (phylo4vcov-class), 38
phylobase (phylobase-package), 3
phylobase-package, 3
phylobase-options, 7, 35
phylobubbles, 36, 50
phylog, 6, 37, 38
phylog-class, 37
phylomat-class, 38
phyloXXX, 37, 39, 40, 41, 49
plot, ANY, ANY-method (treePlot-methods), 49
plot, pdata, missing-method (treePlot-methods), 49
plot, phylo4, missing-method (treePlot-methods), 49
plotOneTree, 40, 40
print, phylo4-method (phylo4-display), 21
print, phylo4d-method (phylo4d-display), 33
printphylo4, 41
prune (subset-methods), 44
prune, phylo4-method (subset-methods), 44
prune, phylo4d-method (subset-methods), 44
prune-methods (subset-methods), 44
ptypes (pdata-class), 18
readNCL (Import Nexus and Newick files), 13
readNewick (Import Nexus and Newick files), 13
readNexus (Import Nexus and Newick files), 13
reorder, phylo-method (reorder-methods), 43
reorder, phylo4-method (reorder-methods), 43
reorder, phylo4d-method (reorder-methods), 43
reorder-methods, 43
reorder.phylo, 44
rootEdge (phylo4-accessors), 19
rootEdge, phylo4-method (phylo4-accessors), 19
rootEdge-methods (phylo4-accessors), 19
rootNode (phylo4-accessors), 19
rootNode, phylo4-method (phylo4-accessors), 19
rootNode-methods (phylo4-accessors), 19
rootNode<-, phylo4-accessors), 19
shortestPath (treewalk), 51
show, phylo4-method (phylo4-display), 21
show, phylo4d-method (phylo4d-display), 33
siblings, 44
siblings (treewalk), 51
subset, phylo4-method (subset-methods), 44
subset, phylo4d-method (subset-methods), 44
subset-methods, 44
sumEdgeLength (treewalk), 51
sumEdgeLength, phylo4-method (treewalk), 51
summary, phylo4-method (phylo4-display), 21
summary, phylo4d-method (phylo4d-display), 33
tail, phylo4-method (phylo4-display), 21
tail, phylo4d-method (phylo4d-display), 33
tbind (multiPhylo-class), 15
tdata, 4, 16, 47
tdata, phylo4d-method (tdata), 47
tdata-method (tdata), 47
tdata <- (tdata), 47
tdata <-, phylo4d, ANY-method (tdata), 47
tdata <-, phylo4d-method (tdata), 47
tip.data.plot, 48
tipData (tdata), 47
tipData, phylo4d-method (tdata), 47
tipData-method (tdata), 47
tipData <- (tdata), 47
tipData <-, phylo4d, ANY-method (tdata), 47
tipData <-, phylo4d-method (tdata), 47
tipLabels (phylo4-labels), 23
tipLabels, phylo4-method (phylo4-labels), 23
tipLabels-methods (phylo4-labels), 23
tipLabels <- (phylo4-labels), 23
tipLabels <-, phylo4, character-method (phylo4-labels), 23
tipLabels <-, phylo4d, character-method (phylo4-labels), 23
treePlot (treePlot-methods), 49
treePlot, phylo4, phylo4d-method (treePlot-methods), 49
treePlot-method (treePlot-methods), 49
treewalk, 51
validObject, 7