Package 'phylobase'

January 30, 2024

Type Package

Title Base Package for Phylogenetic Structures and Comparative Data

Version 0.8.12

Imports ade4, ape (>= 3.0), Rcpp (>= 0.11.0), rncl (>= 0.6.0), grid, methods, stats, RNeXML

LinkingTo Rcpp

Suggests MASS, testthat (>= 0.8.1), knitr, rmarkdown

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)

URL https://github.com/fmichonneau/phylobase

BugReports https://github.com/fmichonneau/phylobase/issues

LazyData true

Collate 'oldclasses-class.R' 'internal-constructors.R'
'phylo4-methods.R' 'RcppExports.R' 'checkdata.R'
'phylo4-class.R' 'getNode-methods.R' 'formatData.R'
'phylo4d-class.R' 'phylo4d-methods.R' 'MRCA-methods.R'
'addData-methods.R' 'ancestors.R' 'phylo4-accessors.R'
'root-methods.R' 'nodeId-methods.R' 'edgeLength-methods.R'
'setAs-methods.R' 'extractTree.R' 'labels-methods.R'
'multiphylo4-class.R' 'pdata.R' 'phylo4d-accessors.R'
'phylobase-package.R' 'phylobase.options.R' 'phylomats-class.R'
'print-methods.R' 'readNCL.R' 'reorder-methods.R'
'shortestPath-methods.R' 'subset-methods.R' 'summary-methods.R'
'tbind.R' 'tdata-methods.R' 'treePlot.R' 'treestruc.R' 'zzz.R'

VignetteBuilder knitr

RoxygenNote 7.3.1

Encoding UTF-8

NeedsCompilation yes

Repository CRAN

Date/Publication 2024-01-30 00:20:03 UTC

R **topics documented:**

Index [62]

---

phylobase-package   *Utilities and Tools for Phylogenetics*

Description

Base package for phylogenetic structures and comparative data.

Details

phylobase provides a set of functions to associate and manipulate phylogenetic information and data about the species/individuals that are in the tree.

phylobase intends to be robust, fast and efficient. We hope other people use the data structure it provides to develop new comparative methods in R.

With phylobase it is easy to ensure that all your data are represented and associated with the tips or the internal nodes of your tree. phylobase provides functions to:

- prune (subset) your trees, find ancestor(s) a descendant(s)
- find the most common recent ancestor of 2 nodes (MRCA)
- calculate the distance of a given node from the tip or between two nodes in your tree
- robust functions to import data from NEXUS and Newick files using the NEXUS Class Library (https://github.com/mtholder/ncl/)

History

phylobase was started during a Hackathlon at NESCent on December 10-14 2007.

Peter Cowan was a Google Summer of Code fellow in 2008 and developed all the code for plotting.

In December 2008, a mini-virtual Hackathlon was organized to clean up and make the code more robust.

In the spring and summer of 2009, Jim Regetz made several contributions that made the code faster (in particular with the re-ordering parts), found many bugs, and wrote most of the testing code.

phylobase was first released on CRAN on November 1st, 2009 with version 0.5.

Since then, several releases have followed adding new functionalities: better support of NEXUS files, creation of phylobase.options() function that controls the phylo4 validator, rewrite of the validator in C++.

Starting with 0.6.8, Francois Michonneau succeeds to Ben Bolker as the maintainer of the package.

More Info

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

See Also

Useful links:

- <https://github.com/fmichonneau/phylobase>
- Report bugs at <https://github.com/fmichonneau/phylobase/issues>

---

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

```
addData(x, ...)

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

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

Arguments

x	a phylo4 or a phylo4d object
...	additional arguments to control how matching between data and tree (see Details section of [phylo4d-methods](#) for more details).
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?

Details

Rules for matching data to tree nodes are identical to those used by the [phylo4d-methods](#) 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-methods](#) 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)
```

---

ancestor	*Tree traversal and utility functions*

Description

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

Usage

```
ancestor(phy, node)

children(phy, node)

descendants(phy, node, type = c("tips", "children", "all", "ALL"))

siblings(phy, node, include.self = FALSE)

ancestors(phy, node, type = c("all", "parent", "ALL"))
```

Arguments

phy	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") or all descendant nodes including self ("ALL").
include.self	whether to include self in list of siblings

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.

Value

ancestors 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

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 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)

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"))
```

---

checkPhylo4   *Validity checking for phylo4 objects*

Description

Basic checks on the validity of S4 phylogenetic objects

Usage

```
checkPhylo4(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 https://emmanuelparadis.github.io/misc/FormatTreeR.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; the [phylo4d-methods](#) 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.

---

edges   *Edges accessors*

Description

Access or modify information about the edges.

Usage

```
edges(x, ...)

## S4 method for signature 'phylo4'
edges(x, drop.root = FALSE)
```

```
edgeOrder(x, ...)

## S4 method for signature 'phylo4'
edgeOrder(x)

internalEdges(x)

## S4 method for signature 'phylo4'
internalEdges(x)

terminalEdges(x)

## S4 method for signature 'phylo4'
terminalEdges(x)
```

Arguments

x	a `phylo4` or `phylo4d` object.
...	Optional arguments used by specific methods. (None used at present).
drop.root	logical (default FALSE), should the edge connecting the root be included in the edge matrix?

Value

edges returns the edge matrix that represent the ancestor-descendant relationships among the nodes of the tree.

edgeOrder returns the order in which the edge matrix is in.

internalEdges returns a logical vector indicating internal edges (edges that connect an internal node to another). This vector is named with the `edgeId`.

terminalEdges returns a logical vector indicating terminal edges (edges that connect an internal node to a tip). This vector is named with the `edgeId`

Author(s)

Ben Bolker, Francois Michonneau, Thibaut Jombart

See Also

reorder, edgeId

Examples

```
data(geospiza)
edges(geospiza)
edgeOrder(geospiza)
geoPost <- reorder(geospiza, "postorder")
edgeOrder(geoPost)
## with a binary tree this should always be true
identical(!terminalEdges(geospiza), internalEdges(geospiza))
```

extractTree   *Get tree from tree+data object*

Description

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

Usage

```
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-methods](#), [phylo4d-methods](#), [coerce-methods](#) for translation functions.

Examples

```
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)
```

formatData   *Format data for use in phylo4d objects*

Description

Associates data with tree nodes and applies consistent formatting rules.

Usage

```
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 rownames 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=="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-methods constructor, the phylo4d 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

Format

geospiza 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

```
getNode(
  x,
  node,
  type = c("all", "tip", "internal"),
  missing = c("warn", "OK", "fail")
)

## S4 method for signature 'phylo4'
getNode(
  x,
  node,
  type = c("all", "tip", "internal"),
  missing = c("warn", "OK", "fail")
)

getEdge(
  x,
  node,
  type = c("descendant", "ancestor"),
  missing = c("warn", "OK", "fail")
)

## S4 method for signature 'phylo4'
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

list("getNode")

                    returns a named integer vector of node IDs, in the order of input nodes if pro-
                    vided, otherwise in nodeId order

list("getEdge")

                    returns a named character vector of edge IDs, in the order of input nodes if
                    provide, otherwise in nodeId order

list("nodeId")   returns an unnamed integer vector of node IDs, in ascending order

list("getEdge")

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

Examples

```
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"))
edges(geospiza)[c(26, 1, 11),]

## quickly get all tip node IDs and tip edge IDs
nodeId(geospiza, "tip")
edgeId(geospiza, "tip")
```

---

hasEdgeLength   *edgeLength methods*

Description

These functions give information about and allow replacement of edge lengths.

Usage

```
hasEdgeLength(x)

## S4 method for signature 'phylo4'
hasEdgeLength(x)

edgeLength(x, ...)

## S4 method for signature 'phylo4'
edgeLength(x, node)

edgeLength(x, use.names = TRUE, ...) <- value

## S4 replacement method for signature 'phylo4'
```

```
edgeLength(x, use.names = TRUE, ...) <- value

depthTips(x)

## S4 method for signature 'phylo4'
depthTips(x)

nodeDepth(x, node)

## S4 method for signature 'phylo4'
nodeDepth(x, node)

nodeHeight(x, node, from)

## S4 method for signature 'phylo4'
nodeHeight(x, node, from = c("root", "all_tip", "min_tip", "max_tip"))

sumEdgeLength(x, node)

## S4 method for signature 'phylo4'
sumEdgeLength(x, node)

isUltrametric(x, tol = .Machine$double.eps^0.5)

## S4 method for signature 'phylo4'
isUltrametric(x, tol = .Machine$double.eps^0.5)
```

Arguments

x	a phylo4 or phylo4d object.
...	optional arguments (none used at present).
node	optional numeric or character vector indicating the nodes for which edge
use.names	should the the name attributes of value be used to match the length to a given edge.
value	a numeric vector indicating the new values for the edge lengths
from	The point of reference for calculating the height of the node. root calculates the distance between the root of the tree and the node. all_tip return the distance between the node and all the tips descending from it. min_tip the distance between the node and its closest tip. max_tip the distance between the node and its farther tip. min_tip and max_tip will be identical if the tree is ultrametric. If more than one tip is equidistant from the node, the tip with the lowest node id will be returned.
tol	the tolerance to decide whether all the tips have the same depth to test if the tree is ultrametric. Default is .Machine$double.eps^0.5.

Details

The edgeLength function returns the edge length in the same order as the edges in the matrix.

Value

> **hasEdgeLength** whether or not the object has edge lengths (logical)
>
> **edgeLength** a named vector of the edge length for the object
>
> **isUltrametric** whether or not the tree is ultrametric (all the tips are have the same depth (distance from the root) (logical)
>
> **sumEdgeLength** the sum of the edge lengths for a set of nodes (intended to be used with `ancestors` or `descendants`)
>
> **nodeHeight** the distance between a node and the root or the tips. The format of the result will depend on the options and the number of nodes provided, either a vector or a list.
>
> **nodeDepth** Deprecated, now replaced by `nodeHeight`. A named vector indicating the "depth" (the distance between the root and a given node).
>
> **depthTip** Deprecated, now replaced by `nodeHeight`.

See Also

`ancestors`, `descendants`, `.Machine` for more information about tolerance.

Examples

```
data(geospiza)
hasEdgeLength(geospiza) # TRUE
topoGeo <- geospiza
edgeLength(topoGeo) <- NULL
hasEdgeLength(topoGeo)  # FALSE

edgeLength(geospiza)[2]       # use the position in vector
edgeLength(geospiza)["16-17"] # or the name of the edge
edgeLength(geospiza, 17)      # or the descendant node of the edge

## The same methods can be used to update an edge length
edgeLength(geospiza)[2] <- 0.33
edgeLength(geospiza)["16-17"] <- 0.34
edgeLength(geospiza, 17) <- 0.35

## Test if tree is ultrametric
isUltrametric(geospiza)   # TRUE
## indeed all tips are at the same distance from the root
nodeHeight(geospiza, nodeId(geospiza, "tip"), from="root")
## compare distances from tips of two MRCA
nodeHeight(geospiza, MRCA(geospiza, c("pallida", "psittacula")), from="min_tip")
nodeHeight(geospiza, MRCA(geospiza, c("fortis", "difficilis")), from="min_tip")
## or the same but from the root
nodeHeight(geospiza, MRCA(geospiza, c("pallida", "psittacula")), from="root")
nodeHeight(geospiza, MRCA(geospiza, c("fortis", "difficilis")), from="root")
```

---

hasSingle   *Test trees for polytomies, inline nodes (singletons), or reticulation*

Description

Methods to test whether trees have (structural) polytomies, inline nodes (i.e., nodes with a single descendant), or reticulation (i.e., nodes with more than one ancestor). `hasPoly` only check for structural polytomies (1 node has more than 2 descendants) and not polytomies that result from having edges with a length of 0.

Usage

```
hasSingle(object)

## S4 method for signature 'phylo4'
hasSingle(object)

hasRetic(object)

## S4 method for signature 'phylo4'
hasRetic(object)

hasPoly(object)

## S4 method for signature 'phylo4'
hasPoly(object)
```

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

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

---

hasTipData	*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

```
hasTipData(x)

## S4 method for signature 'phylo4d'
hasTipData(x)

hasNodeData(x)

## S4 method for signature 'phylo4d'
hasNodeData(x)

nData(x)

## S4 method for signature 'phylo4d'
nData(x)
```

Arguments

x   a phylo4d object

Details

nData tests for the presence of data associated with the object.

hasTipData and hasNodeData tests for the presence of data associated with the tips and the internal nodes respectively. The outcome of the test is based on row names of the data frame stored in the data slot. If no rows have names from the set nodeId(x, "tip"), then hasTipData returns FALSE. Likewise, if no rows have names from the set nodeId(x, "internal"), then hasNodeData returns FALSE.

Value

nData returns the number of datasets (i.e., columns) associated with the object.

hasTipData, hasNodeData return TRUE or FALSE depending whether data associated with the tree are associated with either tips or internal nodes respectively.

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-methods constructor and phylo4d class.

Examples

```
data(geospiza)
nData(geospiza)       ## 5
hasTipData(geospiza)  ## TRUE
hasNodeData(geospiza) ## FALSE
```

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, phylo4d or data.frame object.

Usage

```
readNCL(
  file,
  simplify = FALSE,
  type = c("all", "tree", "data"),
  spacesAsUnderscores = TRUE,
  char.all = FALSE,
  polymorphic.convert = TRUE,
  levels.uniform = FALSE,
  quiet = TRUE,
  check.node.labels = c("keep", "drop", "asdata"),
```

```
    return.labels = TRUE,
    file.format = c("nexus", "newick"),
    check.names = TRUE,
    convert.edge.length = FALSE,
    ...
)

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,
  ...
)

readNewick(
  file,
  simplify = FALSE,
  quiet = TRUE,
  check.node.labels = c("keep", "drop", "asdata"),
  convert.edge.length = FALSE,
  ...
)
```

Arguments

file	a NEXUS file for `readNexus` or a file that contains Newick formatted trees for `readNewick`.
simplify	If TRUE, if there are multiple trees in the file, only the first one is returned; otherwise a list of `phylo4(d)` objects is returned if the file contains multiple trees.
type	Determines which type of objects to return, if present in the file (see Details).
spacesAsUnderscores	In the NEXUS file format white spaces are not allowed in taxa labels and are represented by underscores. Therefore, NCL converts underscores found in taxa labels in the NEXUS file into white spaces (e.g. `species_1` will become `"species 1"`. If you want to preserve the underscores, set as TRUE, the default).
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.
`file.format`	character indicating the format of the specified file (either "newick" or "nexus"). It's more convenient to just use `readNexus` or `readNewick`.
`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 using 'make.names'.
`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 import trees with negative branch lengths.
`...`	Additional arguments to be passed to phylo4 or phylo4d constructor (see Details)

Details

`readNewick` reads a Newick file and outputs a `phylo4` or `phylo4d` object.

`readNexus` is used internally by both `readNexus` and `readNewick` to extract data held in a tree files, specifically in NEXUS files from DATA, CHARACTER or TREES blocks.

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. The `phylo4d` constructor will be used with `type="all"`, or if the option `check.node.labels="asdata"` is invoked.

`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:

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 is thus equivalent to the value drop.

For both readNexus and readNewick, additional arguments can be passed to the constructors such as annote, missing.data or extra.data. See the 'Details' section of phylo4d-methods 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. 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.

Author(s)

Brian O'Meara, Francois Michonneau, Derrick Zwickl

See Also

the phylo4d class, the phylo4 class

---

isRooted   *Methods to test, access (and modify) the root of a phylo4 object.*

Description

Methods to test, access (and modify) the root of a phylo4 object.

Usage

```
isRooted(x)

## S4 method for signature 'phylo4'
isRooted(x)

rootNode(x)

## S4 method for signature 'phylo4'
rootNode(x)

rootNode(x) <- value

## S4 replacement method for signature 'phylo4'
rootNode(x) <- value
```

Arguments

x	a phylo4 or phylo4d object.
value	a character string or a numeric giving the new root.

Value

isRooted logical whether the tree is rooted

rootNode the node corresponding to the root

Author(s)

Ben Bolker, Francois Michonneau

Examples

```
data(geospiza)
isRooted(geospiza)
rootNode(geospiza)
```

MRCA   *MRCA*

Description

Most Recent Common Ancestor (MRCA) of 2 or more nodes.

Usage

```
MRCA(phy, ...)

## S4 method for signature 'phylo4'
MRCA(phy, ...)

## S4 method for signature 'phylo'
MRCA(phy, ...)
```

Arguments

phy	a phylogenetic tree in phylo4, phylo4d or phylo format.
...	a vector of nodes

Details

Given some nodes (i.e., tips and/or internal), this function returns the node corresponding to the most recent common ancestor.

If phy is a phylo4 or phylo4d object, the nodes can contain both numeric or character values that will be used by getNode to retrieve the correct node. However, if phy is a phylo object, the nodes must be a numeric vector.

With phylo4 and phylo4d objects, if a single node is provided, it will be returned.

Value

the node corresponding to the most recent common ancestor

Examples

```
data(geospiza)
MRCA(geospiza, 1, 5)
MRCA(geospiza, "fortis", 11)
MRCA(geospiza, 2, 4, "fusca", 3)
geo <- as(geospiza, "phylo")
MRCA(geo, c(1,5))
```

---

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.

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

---

nodeId   *nodeId methods*

Description

These functions gives the node (nodeId) or edge (edgeId) identity.

Usage

```
nodeId(x, type = c("all", "tip", "internal", "root"))

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

edgeId(x, type = c("all", "tip", "internal", "root"))

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

Arguments

x   a phylo4 or phylo4d object.

type   a character vector indicating which subset of the nodes or edges you are inter-
                ested in.

Details

nodeId returns the node in ascending order, and edgeId in the same order as the edges are stored
in the edge matrix.

Value

nodeId an integer vector indicating node numbers

edgeId a character vector indicating the edge identity

Examples

```
data(geospiza)
identical(nodeId(geospiza, "tip"), 1:nTips(geospiza))
nodeId(geospiza, "internal")
edgeId(geospiza, "internal")
nodeId(geospiza, "root")
```

---

nTips   *nTips, nNodes, nEdges*

Description

Number of tips, nodes and edges found in a tree.

Usage

```
nTips(x)

## S4 method for signature 'phylo4'
nTips(x)

## S4 method for signature 'phylo'
nTips(x)

nNodes(x)

## S4 method for signature 'phylo4'
nNodes(x)

nEdges(x)

## S4 method for signature 'phylo4'
nEdges(x)
```

Arguments

x   a phylo4 or phylo4d object

Details

Function to return the number of tips, nodes and edges found in a tree in the phylo4 or phylo4d format.

Value

a numeric vector indicating the number of tips, nodes or edge respectively.

---

owls4   *'Owls' data from ape*

Description

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

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

```
pdata(data, type, comment, metadata)
```

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

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", ...)`.

Author(s)

Ben Bolker

---

phylo4-class *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](#).

Author(s)

Ben Bolker, Thibaut Jombart

See Also

The [phylo4-methods](#) constructor, the [checkPhylo4](#) function to check the validity of phylo4 objects. See also the [phylo4d-methods](#) constructor and the [phylo4d](#) class.

---

phylo4-labels   *Labels for phylo4/phylo4d objects*

Description

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

Usage

```
labels(object, ...)

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

labels(x, type, use.names, ...) <- value

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

hasDuplicatedLabels(x, type)

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

hasNodeLabels(x)

## S4 method for signature 'phylo4'
hasNodeLabels(x)

nodeLabels(x)

## S4 method for signature 'phylo4'
nodeLabels(x)

nodeLabels(x, ...) <- value

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

tipLabels(x)

## S4 method for signature 'phylo4'
tipLabels(x)

tipLabels(x, ...) <- value

## S4 replacement method for signature 'phylo4'
```

```
tipLabels(x, ...) <- value

hasEdgeLabels(x)

## S4 method for signature 'phylo4'
hasEdgeLabels(x)

edgeLabels(x)

## S4 method for signature 'phylo4'
edgeLabels(x)

edgeLabels(x, ...) <- value

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

Arguments

`object`	a phylo4 or phylo4d object.
`...`	additional optional arguments (not in use)
`type`	which type of labels: `all` (tips and internal nodes), `tip` (tips only), `internal` (internal nodes only).
`x`	a phylo4 or phylo4d object.
`use.names`	should the names of the vector used to create/update labels be used to match the labels? See Details for more information.
`value`	a vector of class `character`, 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.

Value

labels in ascending order.

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

Author(s)

Ben Bolker, Peter Cowan, Steve Kembel, Francois Michonneau

Examples

```
data(geospiza)

## Return labels from geospiza
tipLabels(geospiza)

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

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

## naming the labels
names(ndLbl) <- nodeId(geospiza, "internal")

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

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

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

---

phylo4-methods   *Create a phylogenetic tree*

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

```
phylo4(x, ...)

phylo4_orderings

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

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

## S4 method for signature 'nexml'
phylo4(x)
```

Arguments

x	a matrix of edges or an object of class phylo (see above)
...	optional arguments (none used at present).
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)
annote	any additional annotation data to be passed to the new object
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.
edge	A numeric, two-column matrix with as many rows as branches in the phylogeny.

Format

An object of class character of length 5.

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.

phylobase also allows to create phylo4 objects using the function phylo4() from objects of the classes: phylo (from ape), and nexml (from RNeXML).

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. See also the phylo4d-methods 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-class   *phylo4d class*

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.

Author(s)

Ben Bolker, Thibaut Jombart

See Also

[coerce-methods](#) for translation functions. The [phylo4d-methods](#) constructor. See also the [phylo4-methods](#) constructor, the [phylo4](#) class, and the [checkPhylo4](#) function to check the validity of phylo4 trees.

Examples

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

---

phylo4d-methods   *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

```
phylo4d(x, ...)

## 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",
  annote = list(),
  ...
)

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

## S4 method for signature 'phylo4d'
phylo4d(x, ...)

## S4 method for signature 'nexml'
phylo4d(x)
```

Arguments

x	an object of class `phylo4`, `phylo`, `nexml` or a matrix of edges (see above)
...	further arguments to 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 (see Details).
`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)
`annote`	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).

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 (listed in the Details section); these include positional matching, matching exclusively on node labels, and matching based on a column of data rather than on row names.

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 if you use duplicated labels) 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 uniqeness may lead to surprising results, so this practice should be avoided.

This is the list of additional arguments that can be used to control matching between the tree and the data:

match.data (logical) should the rownames of the data frame be used to be matched against tip and internal node identifiers?

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 iff `label.type=="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

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)`.

Value

An object of class [phylo4d](#).

Methods

x = "phylo4" merges a tree of class `phylo4` with a data.frame into a `phylo4d` object

x = "matrix" merges a matrix of tree edges similar to the edge slot of a `phylo4` 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; phylo4 class and phylo4 constructor.

Examples

```
treeOwls <- "((Strix_aluco:4.2,Asio_otus:4.2):3.1,Athene_noctua:7.3);"
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$tree
geoData <- geospiza_raw$data

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

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

### 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 = geoData, node.data = rNodeData)

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

(matchData1 <- phylo4d(trGeo, tip.data=tDt, node.data=nDt, merge.data=FALSE))
(matchData2 <- phylo4d(trGeo, tip.data=tDt, node.data=nDt, 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'))

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)
```

---

phylobase.options   *Set or return options of phylobase*

Description

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

Usage

```
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.

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

```
phylobubbles(
  type = type,
  place.tip.label = "right",
  show.node.label = show.node.label,
  rot = 0,
  edge.color = edge.color,
  node.color = node.color,
```

```
    tip.color = tip.color,
    edge.width = edge.width,
    newpage = TRUE,
    ...,
    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
`show.node.label`	
	A logical indicating whether internal node labels should be plotted
`rot`	The number of degrees that the plot should be rotated
`edge.color`	A vector of colors for the tree edge segments
`node.color`	A vector of colors for the coloring the nodes
`tip.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 out put from the phyloXXYY function
`square`	Logical indicating whether the plot 'bubbles' should be squares
`grid`	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.
```

---

phylomat-class   *matrix classes for phylobase*

Description

Classes representing phylogenies as matrices

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 im-
plicit) with different meanings depending on the type (variance-covariance matrix, distance matrix,
etc.).

Author(s)

Ben Bolker

Examples

```
tree_string <- "(((Strix_aluco:4.2,Asio_otus:4.2):3.1,Athene_noctua:7.3):6.3,Tyto_alba:13.5);"
tree.owls <- ape::read.tree(text=tree_string)
o2 <- as(tree.owls,"phylo4")
ov <- as(o2,"phylo4vcov")
o3 <- as(ov,"phylo4")
## these are not completely identical, but are
## topologically identical ...

## edge matrices are in a different order:
## cf. edges(o2) and edges(o3)
## BUT the edge matrices are otherwise identical
o2edges <- edges(o2)
o3edges <- edges(o3)
identical(o2edges[order(o2edges[,2]),],
          o3edges[order(o3edges[,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.
```

---

phyloXXYY   *Calculate node x and y coordinates*

Description

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

Usage

```
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.

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 <- phyloXXYY(geospiza)
plot(coor$xx, coor$yy, pch = 20)
```

---

plotOneTree   *Plot a phylo4 object*

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 [phyloXXYY](#) 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
edge.width	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

```
library(grid)
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()
```

---

print	*print a phylogeny*

Description

Prints a phylo4 or phylo4d object in data.frame format with user-friendly column names

Usage

```
print(x, ...)

## S4 method for signature 'phylo4'
print(x, edgeOrder = c("pretty", "real"), printall = TRUE)

show(object)

## S4 method for signature 'phylo4'
show(object)
```

```
names(x)

## S4 method for signature 'phylo4'
names(x)

head(x, ...)

## S4 method for signature 'phylo4'
head(x, n = 20)

tail(x, ...)

## S4 method for signature 'phylo4'
tail(x, n = 20)
```

Arguments

x	a `phylo4` tree or `phylo4d` tree+data object
...	optional additional arguments (not in use)
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
object	a `phylo4` or `phylo4d` object
n	for head() and tail(), the number of lines to print

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 edge matrix.
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

```
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"))
treedata <- phylo4d(tree, tip.data)
plot(treedata)
print(treedata)
```

---

reorder-methods	*reordering trees within phylobase objects*

Description

Methods for reordering trees into various traversal orders

Usage

```
reorder(x, ...)

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

Arguments

x	a phylo4 or phylo4d object
...	additional optional elements (not in use)
order	The desired traversal order; currently only "preorder" and "postorder" are allowed for phylo4 and phylo4d 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.

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.

Note

The `preorder` parameter corresponds to `cladewise` in the ape package, and `postorder` corresponds (almost) to `pruningwise`.

Author(s)

Peter Cowan, Jim Regetz

See Also

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

Examples

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

---

setAs	*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`), RNeXML object (`nexml`), ade4 objects (`phylog`, *now deprecated*), and to `data.frame` representation.

Usage

```
as(object, class)
```

Author(s)

Ben Bolker, Thibaut Jombart, Marguerite Butler, Steve Kembel, Francois Michonneau

See Also

generic `as`, `phylo4-methods`, `phylo4d-methods`, `extractTree`, nexml class from the RNeXML package, `phylog` from the ade4 package and `as.phylo` from the ape package.

Examples

```
tree_string <- "(((Strix_aluco:4.2,Asio_otus:4.2):3.1,Athene_noctua:7.3):6.3,Tyto_alba:13.5);"
tree.owls <- ape::read.tree(text=tree_string)
## 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")
```

---

shortestPath   *shortestPath-methods*

Description

Finds the shortest path between two nodes in a tree

Usage

```
shortestPath(x, node1, node2)

## S4 method for signature 'phylo4'
shortestPath(x, node1, node2)

## S4 method for signature 'phylo'
shortestPath(x, node1, node2)
```

Arguments

x	a tree in the phylo4, phylo4d or phylo format
node1	a numeric or character (passed to getNode) indicating the beginning from which the path should be calculated.
node2	a numeric or character (passed to getNode) indicating the end of the path.

Details

Given two nodes (i.e, tips or internal nodes), this function returns the shortest path between them (excluding node1 and node2 as a vector of nodes.

Value

a vector of nodes indcating the shortest path between 2 nodes

See Also

getNode

---

subset-methods   *Methods for creating subsets of phylogenies*

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

```
subset(x, ...)

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

x[i, ...]

## S4 method for signature 'phylo4,character,missing,missing'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'phylo4,numeric,missing,missing'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'phylo4,logical,missing,missing'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'phylo4,missing,missing,missing'
```

```
x[i, j, ..., drop = TRUE]

## S4 method for signature 'phylo4d,ANY,character,missing'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'phylo4d,ANY,numeric,missing'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'phylo4d,ANY,logical,missing'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'phylo4,ANY,ANY,ANY'
x[i, j, ..., drop = TRUE]

prune(x, ...)

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

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

Arguments

x	an object of class ″phylo4″ or ″phylo4d″
...	optional additional parameters (not in use)
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
i	([method) An index vector indicating tips to include
j	([method, phylo4d only) An index vector indicating columns of node/tip data to include
drop	(not in use: for compatibility with the generic method)
trim.internal	A logical specifying whether to remove internal nodes that no longer have tip descendants in the subset tree

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 nodeId(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 correspondance 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)))
```

---

summary-methods   *Summary for phylo4/phylo4d objects*

Description

Summary of information for the tree (phylo4 only) and/or the associated data (phylo4d).

Usage

```
summary(object, ...)

## S4 method for signature 'phylo4'
summary(object, quiet = FALSE)

## S4 method for signature 'phylo4d'
summary(object, quiet = FALSE)

nodeType(object)

## S4 method for signature 'phylo4'
nodeType(object)
```

Arguments

object	a phylo4d object
...	optional additional elements (not in use)
quiet	Should the summary be displayed on screen?

Value

The `nodeType` method returns named vector which has the type of node (internal, tip, root) for value, and the node number for name

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

`list("name")`	the name of the object
`list("nb.tips")`	
	the number of tips
`list("nb.nodes")`	
	the number of nodes
`list("mean.el")`	
	mean of edge lengths
`list("var.el")`	variance of edge lengths (estimate for population)
`list("sumry.el")`	
	summary (i.e. range and quartiles) of the edge lengths
`list("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
`list("sumry.tips")`	
	(optional) summary for the data associated with the tips
`list("sumry.nodes")`	
	(optional) summary for the data associated with the internal nodes

Author(s)

Ben Bolker, Thibaut Jombart, Francois Michonneau

See Also

[phylo4d-methods](#) constructor and [phylo4d](#) class.

Examples

```
tOwls <- "(((Strix_aluco:4.2,Asio_otus:4.2):3.1,Athene_noctua:7.3):6.3,Tyto_alba:13.5);"
tree.owls <- ape::read.tree(text=tOwls)
P1 <- as(tree.owls, "phylo4")
P1
summary(P1)
nodeType(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
```

---

tdata   *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

```
tdata(x, ...)

## S4 method for signature 'phylo4d'
tdata(
  x,
  type = c("all", "tip", "internal"),
  label.type = c("row.names", "column"),
  empty.columns = TRUE
)

tdata(x, ...) <- value

## S4 replacement method for signature 'phylo4d'
tdata(
  x,
  type = c("all", "tip", "internal"),
  merge.data = TRUE,
  clear.all = FALSE,
  ...
) <- value

tipData(x, ...)

## S4 method for signature 'phylo4d'
tipData(x, ...)
```

```
tipData(x, ...) <- value

## S4 replacement method for signature 'phylo4d'
tipData(x, ...) <- value

nodeData(x, ...)

## S4 method for signature 'phylo4d'
nodeData(x, ...)

nodeData(x, ...) <- value

## S4 replacement method for signature 'phylo4d'
nodeData(x, ...) <- value
```

Arguments

x	A phylo4d object
...	For the `tipData` and `nodeData` accessors, further arguments to be used by `tdata`. For the replacement forms, further arguments to be used to control matching between tree and data (see Details section of [phylo4d-methods](#)).
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 options is useful in the case of missing (NA) or non-unique labels.
empty.columns	Should columns filled with NA be returned?
value	a data frame (or object to be coerced to one) to replace the values associated with the nodes specified by the argument type
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?

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-methods](#), [phylo4d](#)

Examples

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

---

tip.data.plot	*Plotting trees and associated data*

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 [phyloXXYY](#) 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

---

treePlot-methods   *Phylogeny plotting*

Description

Plot phylo4 or phylo4d objects, including associated data.

Usage

```
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",
  plot.at.tip = TRUE,
  edge.color = "black",
  node.color = "black",
  tip.color = "black",
  edge.width = 1,
  newpage = TRUE,
  margins = c(1.1, 1.1, 1.1, 1.1),
  ...
)

plot(x, y, ...)

## S4 method for signature 'phylo4,missing'
plot(x, y, ...)
```

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 the each tip of the phylogenetic trees
`plot.at.tip`	should the data plots be at the tip? (logical)
`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
`margins`	number of lines around the plot (similar to par(mar)).
`...`	additional arguments
`x`	A phylo4 or phylo4d object
`y`	(only here for compatibility)

Details

Currently, `treePlot` can only plot numeric values for tree-associated data. The dataset will be subset to only include columns of class `numeric`, `integer` or `double`. If a `phylo4d` object is passed to the function and it contains no data, or if the data is in a format that cannot be plotted, the function will produce a warning. You can avoid this by using the argument `plot.data=FALSE`.

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>, Francois Michonneau

See Also

phylobubbles

Examples

```
## example of plotting two grid plots on the same page
library(grid)
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"))
    treePlot(geotree, newpage=FALSE)
  popViewport()

  pushViewport(viewport(layout.pos.col=2, name="plot2"))
    treePlot(geotree, newpage=FALSE, rot=180)
popViewport(2)
```

Index