Package 'phyext'

July 2, 2014

Type Package

Title An extension of some of the classes in phylobase. Tree objects now support subnodes on branches

Version 0.0.1

Date 2011-01-26

Author J. Conrad Stack <stack@psu.edu>

Maintainer J. Conrad Stack <stack@psu.edu>

Depends methods, phylobase (>= 0.6.2), ape (>= 2.5)

Enhances phylobase

Description This package enhancements to phylobase classes. It provides classes and methods which help uses manipulate branch-annotated trees and provides supports for a few other extra features.

License GPL-3

LazyLoad yes

Collate processNexus.R class-phylo4d_ext.R methods-phylo4d_ext.R setAs-phyext.R phyextPlots.R ci.R zzz.R

Repository CRAN

Repository/R-Forge/Project phyext

Repository/R-Forge/Revision 4

Date/Publication 2011-05-02 09:39:43

NeedsCompilation no

R topics documented:

addSubNode *Add subnode to phylo4d_ext object*

Description

Add a subnode to phylo4d_ext object.

Usage

```
addSubNode(x, anc, dec, position, dataf, pos.is.fraction = FALSE)
```

Arguments

x	a phylo4d_ext object
anc	Node id of the ancestor on the branch on which the subnode is to be added
dec	Node id of the descendant on the branch on which the subnode is to be added
position	The position on the branch where the subnode is to be placed. It is stored as a faction of the branches' total length and is in relation to the ancestral node. In other words, a smaller value will be closer to the ancestral node.
dataf	a data.frame object which should be structurally similar to the data frame of the phylo4d object (data slot). It should contain data entries for the subnode.
pos.is.fraction	Specify whether position argument a fraction of the branch length or an absolute number.

Details

This is the main function in RBrownie for adding subnodes manually. It is used internally when reading in SIMMAP-formatted newick trees and can also be used to edit phylo4d_ext trees.

Value

Returns x with a new subnode added.

See Also

phyext, phyextPlot

collapse.singletons *Collapse singleton nodes*

Description

This function turns any singleton nodes (internal non-root nodes with degree = 2) into subnodes. It uses the function addSubNodes to attach the nodes to a phylo4d_ext object. These subnodes are positioned on the branch relative to the descendant node.

Usage

```
collapse.singletons(phy)
```

Arguments

phy	phylo4 tree with singletons that will potentially be collapsed.

Value

The tree with the singletons collapsed.

Author(s)

J. Conrad Stack

See Also

hasSubNodes

collapse.subnodes *Collapse subnodes into singletons*

Description

Method which converts any existing subnodes into their phylobase equilvalent, singleton nodes (nodes with an in-degree and out-degree equal to one).

Usage

```
collapse.subnodes(x, rm.ex.data = TRUE)
```

Arguments

x	An object of type phylo4d_ext which contains subnodse
rm.ex.data	Should data columns that only exist in subnodes.data be removed (TRUE is recommended).

Details

This functions allows a user to convert from `RBrownie:::phylo4d_ext` objects to `phylobase:::phylo4d` objects without loosing any of the tree's topological information (or data). This is in contrast to the `collapse.singletons` function which removes subnodes and returns a phylo4d tree without singletons or subnodes

Value

An object of class `phylo4d` with singleton nodes where subnodes used to be

Author(s)

J. Conrad Stack

See Also

`collapse.singletons`, `phyext`, `expand.singles`

`collapse.to.singles` *Collapse zero-length branches*

Description

This function is not being used at this point, but works. Sister function of expand.singles -> converts internal nodes with zero-length branches into singletons (in phylo4 format). Will only work if the zero-len branch is connected to a tip.

Usage

```
collapse.to.singles(tree, by.name = NULL)
```

Arguments

`tree`	Phylo4 object
`by.name`	If this is set then tip labels are searched for this pattern and those nodes are collapsed. (see `tipLabels` function in phylobase).

Details

Not currently being used.

Value

A new phylogenetic tree which contains singleton nodes (possibly)

Author(s)

J. Conrad Stack

deep.phy.copy *Copy a tree object slot by slot*

Description

This is mainly an internal function

Usage

```
deep.phy.copy(from, to, value, usedata = TRUE)
```

Arguments

from	An S4 object
to	An S4 object
value	The value to copy
usedata	Should data slot be included?

Details

This function is not being used currently

Value

A copy of from

Author(s)

J. Conrad Stack

expand.singles *Expand singletons*

Description

This function expands singleton nodes into bifurcating nodes with one zero-length branch. This isn't much and was originally used to convert phylo4 objects with singletons into phylo objects ('ape'). This adds new taxa and assigns them a unique identifiier name.

Usage

```
expand.singles(tree, keep.data=FALSE)
```

Arguments

tree	A tree with singleton nodes
keep.data	Should a phyl4d (TRUE) or phylo4 (FALSE) object be returned.

Details

This function isn't used much, but might be used to convert phylo4 objects with singletons to phylo objects with zero-length branches. Might be helpful for writing singletons to file (via ape).

Value

A phylo4 object with the singletons converted.

Author(s)

J. Conrad Stack

---

get.nexus.comments *Get comments from nexus text*

Description

This function reads in a nexus file or uses supplied text and then returns the comments. The comments start with [and end with]. This function may not work with multiline comments; don't expect it to do so.

Usage

```
get.nexus.comments(finput, text = NULL)
```

Arguments

finput	A file which is read to text
text	Character text which can be alternatively supplied

Author(s)

J. Conrad Stack

get.nodenames *Read node names from newick text*

Description

Internal function to help processing SIMMAP newick files.

Usage

```
get.nodenames(newick.txt)
```

Arguments

newick.txt	Character string of newick text

Value

The node names

Author(s)

J. Conrad Stack

get.tree.weights *Get tree weights*

Description

Read the tree weights from a nexus file or text. It looks within the comments before each tree string for the weights.

Usage

```
get.tree.weights(finput,
text = NULL,
starters = c("&W", "&lnP"),
splitchar = "( |=)")
```

Arguments

finput	A file to be read in
text	Alternatively, send it text itself.
starters	The character pattern which indicates the start of the weight (within a comment)
splitchar	The character pattern which separates the 'starters' from the weights themselves

Value

Returns a numeric vector of the weights (if any are avialable).

Author(s)

J. Conrad Stack

`getEmptyDataFrame` *Get empty data.frame*

Description

Get an empty data frame with the same structure and column names of the input data frame

Usage

```
getEmptyDataFrame(x, ...)
```

Arguments

`x`	The data frame whose structure is to be copied
`...`	Arguments passed to data.frame() function

Details

This is mainly an internal function used to generate subnode dataframes.

Value

An empty data.frame

Author(s)

J. Conrad Stack

getSubNodeData *Return subnode data*

Description

Return the subnode data.frame

Usage

```
getSubNodeData(x, colname)
```

Arguments

x	A phylo4d_ext object
colname	The column of the subnode data.frame to return

Value

The data.frame or column

Author(s)

J. Conrad Stack

getSubNodeEdgeInds *Return subnode edge indices*

Description

The function gets the node indices on which each subnode is positioned on.

Usage

```
getSubNodeEdgeInds(x)
```

Arguments

x	A phylo4d_ext object

Details

Mainly an internal function

Author(s)

J. Conrad Stack

getSubNodePosish *Return the subnode position*

Description

This function returns the position of subnodes as a two-column matrix. The two-columns are used for now because the position will eventually be represented as an interval to reflect the uncertainty of node placement.

Usage

```
getSubNodePosish(x)
```

Arguments

x	A phylo4d_ext

Value

A two-column matrix with the positions

Author(s)

J. Conrad Stack

has.block *Check for nexus block*

Description

Check if nexus text contains a certain block. This is mainly an internal function.

Usage

```
has.block(finput, txt = NULL, blockname = "characters2")
```

Arguments

finput	A nexus file
txt	Alternatively, pass the nexus text in directly
blockname	The block name to look for

Value

True if the block is found

Author(s)

J. Conrad Stack

---

has.characters2 *Check for CHARACTERS2 block*

Description

Check for CHARACTERS2 block in a nexus file or text

Usage

```
has.characters2(finput, txt = NULL)
```

Arguments

finput

txt

Details

This uses has.block

Value

True if block exists

Author(s)

J. Conrad Stack

---

has.weights *Check for tree weights*

Description

Check if a file or text contains tree weights

Usage

```
has.weights(finput, text = NULL)
```

Arguments

finput	A nexus file
text	Nexus text

Details

If function is still experimental.

Value

True or false if the weights exist

Author(s)

J. Conrad Stack

`hasData-methods` *Methods for Function hasData in Package 'RBrownie'*

Description

This method checks to see if tree objects have data

Methods

`signature(x = "list")` Calls hasData again on each element of the list

`signature(x = "phylo")` Returns FALSE

`signature(x = "phylo4")` Returns FALSE

`signature(x = "phylo4d")` Check if `ncol(tdata(x))` is equal to zero or not

`hasDataColumn-methods` *Does a tree object contain a specific data column*

Description

This function is used to query whether or not a class (inheriting from 'phylo4d') contains a specific data column defined either by its column index or it's column name

Methods

`signature(x = "phylo4", index = "ANY")` Return FALSE

`signature(x = "phylo4d", index = "character")` Checks whether or not the data slot contains a column with the name `index`

`signature(x = "phylo4d", index = "numeric")` Checks whether or not `index` is less than or equal to `ncol(tdata(x))`

`signature(x = "phylo", index = "ANY")` Return FALSE

hasSubNodes-methods *Does an object contain subnodes?*

Description

Check to see if a 'phylo4d_ext' objects actually contains subnodes

Methods

`signature(x = "list")` Apply this function to each object in the list

`signature(x = "phylo")` Returns FALSE

`signature(x = "phylo4")` Returns FALSE

`signature(x = "phylo4d_ext")` Checks if `subnode.data` contains any columns

is.simmap *Check SIMMAP v1.0*

Description

Check if text of file contains a SIMMAP-formatted (v1.0) newick file

Usage

```
is.simmap(finput = "", text = NULL, vers=c(1.1),quick=TRUE)
```

Arguments

`finput`	A newick file
`text`	Newick file text
`vers`	Which simmap version to use
`quick`	If TRUE, then this function actually tries to explicitly convert the input to the specified simmap vers and catch any errors

Details

This function looks for '' and '' (or '[' and ']' depending on vers) characters in the text which would indicate that it is a SIMMAP-formatted file. Currently, versions 1.0 and 1.1 are supported.

Value

Logical

Author(s)

J. Conrad Stack

match.order *Matching orders*

Description

Internal function for `read.simmap.new`: Match the order of one list ot the order of another

Usage

```
match.order(basevec, cmpvec)
```

Arguments

`basevec`	The vector whose order that is to be matched to
`cmpvec`	The vector whose order needs to be changed

newlabels.v15 *Generate simmap v1.5 labels*

Description

This is a helper function for write.simmap and is chiefly for internal use. It generates new simmap style text labels to be written to a tree string.

Usage

```
newlabels.v15(x, usestate, splitchar)
```

Arguments

`x`	The phylo4d_ext object to generate simmap (1.5) labels for.
`usestate`	Which data column should be used written in the labels. Note: If left as NULL (default), then all columns will be used
`splitchar`	Should always be a comma

Value

A character vector containing simmap-style taxa labels

See Also

`newlabels.v1x`, `write.simmap`

newlabels.v1x *Generate simmap v1.0 or v1.1 labels*

Description

This is a helper function for write.simmap and is chiefly for internal use. It generates old simmap style text labels to be written to a tree string.

Usage

```
newlabels.v1x(x, usestate, splitchar, write.nas = TRUE)
```

Arguments

x	The phylo4d_ext object to generate simmap (1.0 or 1.1) labels for.
usestate	Which data column should be used written in the labels. Note: Only the first will be used
splitchar	Which character should be used to split subnodes (for 1.0 -> ';' and for 1.1 -> ':')
write.nas	Should NA values be written in the string (TRUE) or ignored (FALSE)

Value

A character vector containing simmap-style taxa labels

Author(s)

J. Conrad Stack

See Also

newlabels.v15, write.simmap

nSubNodes *Get number of subnodes*

Description

Return the number of subnodes in a phylo4d_ext object

Usage

```
nSubNodes(x)
```

Arguments

x	A phylo4d_ext object

Value

Return the number of subnodes.

Author(s)

J. Conrad Stack

---

phyext-methods *A function which returns a phylo4d_ext object*

Description

This is a shorthand way of creating phylo4d_ext objects

Methods

`signature(x = "character")` This is not implemented yet.

`signature(x = "list")` Convert a list of phylo,phylo4,or phylo4d objects into a list of phylo4d_ext objects

`signature(x = "phylo")` Convert to phylo4d_ext

`signature(x = "phylo4")` Convert to phylo4d_ext

`signature(x = "phylo4d")` Convert to phylo4d_ext

`signature(x = "phylo4d_ext")` Returns the argument without modification

---

phyextPlot *Plot phylo4d_ext object*

Description

This function plots a phylo4d_ext object. Branches and nodes in different states can be colored. If subnodes exist it also plots these on the tree, showing them on the exact position they actually occur. This plotting function is an extension to treePlot in phylobase. It was created to plot discrete data states on top of a tree.

Usage

```
phyextPlot(x,
states,
states.col,
states.na = "none",
usestate = 1,
plot.subnodes = T,
plot.points = T,
line.widths,
line.types,
...)
```

Arguments

x	The phylo4d_ext object
states	A vector containing all the discrete states to show on the plot.
states.col	The colors for each state. The branches are painted these colors
states.na	If states is missing, then this is used to represent the NAs in the dataset. It is mainly for internal use.
usestate	This is the column index for a discrete data column whose states will be painted onto the tree. The data column specified should contain data for every node in the tree. For any missing data the states.na will be used.
plot.subnodes	Logical specifying whether or not subnodes should be plotted (if they exist).
plot.points	Logical specifying whether or not points should be plotted at each internal node and subnode. This can help indicate where subnodes exist on a branch.
line.widths	Integer. Line widths for each state. Branches in this state will use this line width.
line.types	Integer. Line types for each state. Branches in this state will use this line type.
...	This is passed to the phylobase function treePlot, which is the function that this function uses to place the tree on the canvas.

Details

This is nearly an internal function. It basically uses treePlot (to which you can pass options via ...) to plot a phylogenetic tree (with no continuous data). It then plots discrete data on top of this plot based on the which state each branch is in. The branch is colored based on the state of the descendant node.

Note that treePlot is called first and actually drawn on the canvas before the colored branches are plotted on top. This could lead to some odd plots when the using different line.widths and line.types. To neutralize this effect try passing colors to treePlot through ... which paints the original plot the same color as the background color.

This is the default function for phylo4d_ext.

Author(s)

J. Conrad Stack

phylo4d_ext-class *Class "phylo4d_ext"*

Description

This class extends phylo4d in adding support for subnodes (not singletons) and tree weights.

Objects from the Class

Objects can be created by calls of the form `new("phylo4d_ext", ...)`.

Slots

`subnode.id`: Object of class `"integer"`. The subnode identifier. Not currently being used.

`subnode.data`: Object of class `"data.frame"`. The subnode data frame. This data frame should extend the parent phylo4d class data frame - it extends the parent data frame in the same way the phylo4d_ext extends phylo4d.

`subnode.branch`: Object of class `"matrix"`. This is a two-column matrix. The first column contains the ancestor node Id and the descendant node Id for the two nodes spanning a branch. This copies data directly from the edge slot. It uses the node ids instead of the edge row index because the edge slot is often reordered.

`subnode.pos`: Object of class `"matrix"`. This is the position of the subnode on the branch as a fraction of the branch length. It is the fraction of the branch length starting from the descendant node.

So for the hypothetical branch: ANC <———subnode–> DEC, the subnode position might be 0.25.

This is currently a two column matrix, but the columns should contain the same data. Eventually, these two columns will represent a confidence interval that the subnode might exist within.

`weight`: Object of class `"numeric"`. This is the weight for this tree. The weight is gleaned from nexus comments in a nexus file or set manually.

This is the only piece of data that is not replicated across every member of a phylo4d_ext list. Each tree can have a unique weight.

`data`: Object of class `"data.frame"`. Inherited from phylo4d

`metadata`: Object of class `"list"`. Inherited from phylo4d

`edge`: Object of class `"matrix"`. Inherited from phylo4

`edge.length`: Object of class `"numeric"`. Inherited from phylo4

`label`: Object of class `"character"`. Inherited from phylo4

`edge.label`: Object of class `"character"`. Inherited from phylo4

`order`: Object of class `"character"`. Inherited from phylo4

`annote`: Object of class `"list"`. Inherited from phylo4

Extends

Class "phylo4d", directly. Class "phylo4", by class "phylo4d", distance 2.

Methods

hasWeight `signature(x = "phylo4d_ext")`: Checks whether or not a phylo4d_ext object or list of these objects contain weights.

plot `signature(x = "phylo4d_ext", y = "missing")`: Calls `phyextPlot` with no extra arguments

snbranch `signature(x = "phylo4d_ext")`: Return the subnode.branch matrix

sndata `signature(x = "phylo4d_ext")`: Return the subnode.data data.frame

sndata<- `signature(x = "phylo4d_ext")`: This adds column to the subnode data.frame. Pass the datnames option which will be the column name(s) of the new data columns.

snid `signature(x = "phylo4d_ext")`: Return the subnode.id vector

snposition `signature(x = "phylo4d_ext")`: Return the subnode.pos matrix

weight `signature(x = "phylo4d_ext")`: Return the weight for a phylo4d_ext object or a vector of weights for a list of phylo4d_ext objects.

weight<- `signature(x = "phylo4d_ext")`: Replaces the weight vector.

Author(s)

J. Conrad Stack

References

~put references to the literature/web site here ~

See Also

phylo4d

`read.characters2` *Read CHARACTERS2 block*

Description

This function reads raw text in from the nexus file finput. It then writes the TAXA and `blockname` blocks to a temporary nexus file using CHARACTERS instead of `blockname`. It then re-reads the temporary nexus file using `readNexus(tmpfile,type="data")`.

Usage

```
read.characters2(finput, blockname = "characters2")
```

Arguments

`finput`

`blockname`

Details

This function was created mainly to handle reading CHARACTERS2 nexus blocks. These blocks are used by brownie along with CHARACTERS to separate discrete (factor) data from continuous (numeric) data. It is also used because SIMMAP-formatted (version < 1.5) nexus files tend to cause problems with the nexus class library (which phylobase uses)

Value

A data.frame containing the data from `blockname`.

Author(s)

J. Conrad Stack

See Also

`readNexus`

`read.nexus.block` *Find a block in a nexus file*

Description

Scan a nexus file for a certain block. This is mainly an internal function, but might be used to diagnose problems with nexus formatted files. It doesn't attempt to process the nexus file, just searches the rawtext for blocks

Usage

```
read.nexus.block(finput, txt = NULL, block, rm.comments = F, silent = F)
```

Arguments

`finput`	A nexus file
`txt`	Alternatively, look through this text
`block`	The block name to look for
`rm.comments`	Logical. Should comments be removed? This option is experimental at the moment and does not work for multiline comments.
`silent`	Verbose output?

Details

Eventually, multiline comments will be supported once the read and scanning of nexus files becomes more sophisticated.

Value

A character vector of the text in the `block` block.

Author(s)

J. Conrad Stack

`read.nexus.simmap` *Read a nexus file with SIMMAP-formatted trees (v1.0)*

Description

This function reads the trees from a nexus file where the trees are SIMMAP-formatted (v1.0). It does not read any data blocks, just the trees. If the file contains multiple trees, this function returns them as a list.

Usage

```
read.nexus.simmap(finput = "", text = NULL, vers = NULL, ...)
```

Arguments

`finput`	A nexus file
`text`	Alternatively, read this text.
`vers`	Which version of simmap to try to read.
`...`	All other arguments are passed to internal `read.simmap` function.

Details

This function reads nexus blocks using a combination of internal functions and a few phylobase functions. If simmap newick trees are found then the function `read.simmap` is used otherwise a modified `read.tree` is found.

Value

A list containing either phylo4d objects or phylo4d_ext objects depending on whether or not SIMMAP characters are found.

Author(s)

J. Conrad Stack

read.simmap *Read simmap-formatted newick strings (version 1.0)*

Description

This function takes in a text string which contains a simmap-formatted newick string. ” and ” are used in these strings to indicate subnodes and character states.

Usage

```
read.simmap(file = "", text = NULL, vers = 1.1, as.num = FALSE, ...)
```

Arguments

file	This is not being used
text	The simmap newick string
vers	Which SIMMAP format to use. vers=1.0 uses ’;’ to split up tokens within a simmap block. vers=1.1 uses ’:’. See examples
as.num	Not currently used.
...	Not currently used.

Value

A phylo4d_ext object

Author(s)

J. Conrad Stack

read.simmap.new *Read simmap-formatted newick strings (version 1.5)*

Description

This function takes in a text string which contains a simmap-formatted newick string. ’[’ and ’]’ are used in these strings to indicate subnodes and character states.

Usage

```
read.simmap.new(file = "", text = NULL, specialpatt = character(0))
```

Arguments

file	A file containing a newick string with SIMMAP 1.5 style comments
text	The simmap newick string
specialpatt	A pattern to match against variable names. NOTE: For now, it should almost always be set to '.*' and needs to be further tested before anything else should be tried.

Details

This is still a work in progress.

---

rmdata-methods *rmdata from a tree object*

Description

This is the recommended way of removing data from the data and subnode . data slots of 'phylo4d_ext' and 'brownie' objects. Although this method is defined for both 'brownie' and 'phylo4d_ext' classes, there is not much difference between them.

Methods

signature(x = "brownie", index = "character", subindex = "missing") Enter the column name of the data column to be removed (see colnames(tdata(x)) for valid options). subindex is set to the value of index. This function will attempt to remove that column from both regular and subnodes.

signature(x = "brownie", index = "numeric", subindex = "ANY") Enter the column index of the data column to be removed. When subindex is ANY, it is ignored (except in the cases below) and set to the value of index

signature(x = "brownie", index = "numeric", subindex = "missing") Enter the column index of the data column to be removed (seq(ncol(tdata(x))) are valid options). subindex is set to the value of index. This function will attempt to remove that column from both regular and subnodes, regardless of whether their column names match up.

signature(x = "brownie", index = "numeric", subindex = "numeric") Used internally: remove the data column (index) and the subnode data column (subindex) from their respective slots

signature(x = "list", index = "ANY", subindex = "missing") Applys rmdata to each object in a list

signature(x = "phylo4d", index = "character", subindex = "missing") Enter the column name of the data column to be removed. No need to specify a subindex

signature(x = "phylo4d_ext", index = "character", subindex = "missing") Enter the column name of the data column to be removed (see colnames(tdata(x)) for valid options). subindex is set to the value of index. This function will attempt to remove that column from both regular and subnodes.

`signature(x = "phylo4d_ext", index = "numeric", subindex = "missing")` Enter the column index of the data column to be removed (`seq(ncol(tdata(x)))` are valid options). `subindex` is set to the value of `index`. This function will attempt to remove that column from both regular and subnodes, regardless of whether their column names match up.

`signature(x = "phylo4d_ext", index = "numeric", subindex = "numeric")` Used internally: remove the data column (`index`) and the subnode data column (`subindex`) from their respective slots

`signature(x = "phylo4d", index = "numeric", subindex = "missing")` Enter the column index of the data column to be removed. No need to specify a subindex

---

`showSubNodes` *Display subnodes*

Description

Display a phylo4d_ext object. This mainly extends the show function for phylo4 object in that it also displays any subnodes if they exist.

Usage

```
showSubNodes(x)
```

Arguments

`x`	A phylo4d_ext object

Details

Attempts to print out the branches which contain subnodes and where on the branches the subnodes are. This is still a bit experimental.

Value

Prints out the phylo4d_ext object.

Author(s)

J. Conrad Stack

strip *Strip whitespace*

Description

Internal function: strip whitespace from the ends of a string

Usage

```
strip(str, left = TRUE, right = TRUE)
```

Arguments

str	A string
left	Strip white space from left end of string
right	Strip white space from right end of string

tipdate.ci *Get coalescent intervals for a tip-dated tree*

Description

Similar to ape's coalescent intervals, except it works when the tree is not ultrametric. This is being used for now as a helper function for treeHeight

Usage

```
tipdate.ci(tr, show.plot = F)
```

Arguments

tr	A tree in phylo format
show.plot	Should a plot be generated highlighting the coalescent events

Details

This is mainly being used with treeHeights for now, but it is a good way to get the coalescent intervals for a tree. Be aware though that the tree needs to be down-convert to `ape:::phylo` format first.

Value

lineages	The number of extant lineages during this interval
interval.length	The length of each interval
interval.count	The total number of coalescent intervals
total.depth	The height or depth of the tree
I	Indicator for whether or not the event is a coalescent event (1) or a sampling event (0)

Author(s)

J. Conrad Stack

---

treeHeight-methods *Get the height (depth) of a tree*

Description

Get the absolute height (or depth) of a given tree. This function uses tipdate.ci to accomplish this by first converting the argument to class 'phylo' and then calling it with that converted value.

Methods

signature(x = "phylo4") Get the tree height

signature(x = "phylo4d") Get the tree height

signature(x = "phylo4d_ext") Get the tree height

---

validPhylo4d_ext *Validate a phylo4d_ext object*

Description

Internal function used by RBrownie to check whether a valid phylo4d_ext is created with the phyext function

Usage

```
validPhylo4d_ext(object)
```

Arguments

object	The object which you want to check for compatibility with the phylo4d_ext class

Value

TRUE if the object is valid

Author(s)

J. Conrad Stack

`write.nexus.simmap` *Write phylo4d_ext object or list to a nexus file*

Description

This function is similar to `write.nexus` from the ape package. If the there are subnodes in the object then these are written as SIMMAP-formatted (v1.0) strings.

Usage

```
write.nexus.simmap(obj, file = "", translate = TRUE, ...)
```

Arguments

`obj`	A phylo4d_ext object or list of these objects
`file`	The file to output to
`translate`	Should taxa names be translated (not written in every newick string) or not.
`...`	Other arguments which are passed to `write.simmap`

Details

This function will eventually support more subnode output formats other than just SIMMAP v1.0. Work is in progress on this. But for now, that is the only format that the brownie core reads and this function was written mainly to generate brownie files.

It borrows a lot of it's code and structure from the ape function `write.nexus`

Value

Nothing

Author(s)

J. Conrad Stack

See Also

`read.nexus.simmap`, `write.simmap`

`write.simmap` *Write phylo4d_ext tree to newick string*

Description

This function write a phylo4d_ext object to a SIMMAP-formatted (version 1.0 or 1.1) text string. This function is mainly used in conjunction with `write.nexus.simmap`. It is an extension of the `write.tree` function in the ape package.

Usage

```
write.simmap(x, usestate = NULL, file = "", vers = 1.1, ...)
```

Arguments

`x`	A phylo4d_ext object
`usestate`	Which data column should be written in the internal simmap blocks. If NULL, then either the first data column (vers==1.1 || vers==1.0) or all data columns (vers==1.5) are used.
`file`	The file to write to. If not set, then the tree is printed to the screen
`vers`	Which version to use. 1.0 uses the ';' as the internal delimiter which 1.1 uses ':'
`...`	These are passed to write.tree (which is only called if there are no subnodes)

Value

A character string (if `file` is not set)

Author(s)

J. Conrad Stack

See Also

`newlabels.v15`, `newlabels.v1x`, `write.nexus.simmap`

write.simmap.new *Wrapper for write.simmap*

Description

Just a wrapper function for write.simmap which instructs write.simmap to use simmap version 1.5

Usage

```
write.simmap.new(x, usestate = NULL, file = "", ...)
```

Arguments

x	A tree object compatible with write.simmap
usestate	Which state to output? If left as NULL, then all the data columns are used
file	Where out output the simmap style tree string to?
...	Other arguments which are passed on to write.simmap

Author(s)

J. Conrad Stack

See Also

write.simmap

write.simmap.old *Wrapper for write.simmap*

Description

Just a wrapper function for write.simmap which instructs write.simmap to use simmap version 1.1

Usage

```
write.simmap.old(x, usestate = NULL, file = "", ...)
```

Arguments

x	A tree object compatible with write.simmap
usestate	Which state to output? If left as NULL, then the first data column is used
file	Where out output the simmap style tree string to?
...	Other arguments which are passed on to write.simmap

Author(s)

J. Conrad Stack

See Also

write.simmap

Index