Package 'treeman'

April 6, 2018

Type Package

Title Phylogenetic Tree Manipulation Class and Methods

Version 1.1.2

Author D.J. Bennett

Maintainer D.J. Bennett <dominic.john.bennett@gmail.com>

Description S4 class and methods for intuitive and efficient phylogenetic tree manipulation.

License GPL-2

Depends R (>= 3.2.4), methods

Imports plyr, ape, RJSONIO, stringr, bigmemory

Suggests testthat

RoxygenNote 6.0.1

Collate 'calc-methods.R' 'check_methods.R' 'cnvrt-methods.R'
'gen-methods.R' 'get-nd-methods.R' 'get-nds-methods.R'
'get-spcl-methods.R' 'manip-methods.R' 'ndlst-methods.R'
'ndmtrx-methods.R' 'node-declaration.R' 'read-write-methods.R'
'server-methods.R' 'set-methods.R' 'treeman-declaration.R'
'treemen-declaration.R' 'update-methods.R' 'viz-methods.R'
'zzz.R'

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-04-06 12:36:57 UTC

R topics documented:

`treeman-package` *Phylogenetic Tree Manipulation*

Description

Manipulate phylogenetic trees in R simply, intuitively and efficiently with a list-based tree structure.

Details

Package:	treeman
Type:	Package
Version:	1.0
Date:	2015-12-02
License:	GPL-2

Author(s)

D.J. Bennett

Maintainer: D.J. Bennett <dominic.john.bennett@gmail.com>

References

No key references

See Also

https://github.com/DomBennett/treeman/wiki

Examples

```
library(treeman)
tree <- randTree(100)
print(tree)
```

`addClade` *Add clade to tree*

Description

Returns a tree with added clade

Usage

```
addClade(tree, id, clade)
```

Arguments

`tree`	`TreeMan` object
`id`	tip/node ID in tree to which the clade will be added
`clade`	`TreeMan` object

Details

Add a TreeMan object to an existing TreeMan object by specifying an ID at which to attach. If the id specified is an internal node, then the original clade descending from that node will be replaced. Before running, ensure no IDs are shared between the tree and the clade, except for the IDs in the clade of that tree that will be replaced. Note, returned tree will not have a node matrix.

See Also

rmClade, getSubtree, https://github.com/DomBennett/treeman/wiki/manip-methods

Examples

```
library(treeman)
t1 <- randTree(100)
# extract a clade
cld <- getSubtree(t1, 'n2')
# remove the same clade
t2 <- rmClade(t1, 'n2')
# add the clade again
t3 <- addClade(t2, 'n2', cld)
# t1 and t3 should be the same
# note there is no need to remove a clade before adding
t3 <- addClade(t1, 'n2', cld)  # same tree
```

---

addNdmtrx *Add node matrix to a tree*

Description

Return tree with node matrix added.

Usage

```
addNdmtrx(tree, shared = FALSE, ...)
```

Arguments

tree	TreeMan object
shared	T/F, should the bigmatrix be shared? See bigmemory documentation.
...	as.big.matrix() additional arguments

Details

The node matrix makes 'enquiry'-type computations faster: determining node ages, number of descendants etc. But it takes up large amounts of memory and has no impact on adding or removing tips. Note, trees with the node matrix can not be written to disk using the 'serialization format' i.e. with save or saveRDS. The matrix is generated with bigmemory's 'as.big.matrix()'.

See Also

updateSlts, rmNdmtrx, https://cran.r-project.org/package=bigmemory

Examples

```
# library(treeman)
tree <- randTree(10, wndmtrx=FALSE)
summary(tree)
tree <- addNdmtrx(tree)
summary(tree)
```

---

addTip *Add tip to a tree*

Description

Returns a tree with a new tip ID added

Usage

```
addTip(tree, tid, sid, strt_age = NULL, end_age = 0, tree_age = NULL,
  pid = paste0("p_", tid))
```

Arguments

tree	TreeMan object
tid	tip ID
sid	ID of node that will become new tip sisters
strt_age	timepoint at which new tips first appear in the tree
end_age	timepoint at which new tips end appear in the tree, default 0.
tree_age	age of tree
pid	parent ID (default is 'p_' + tid)

Details

User must provide new tip ID, the ID of the node which will become the new tip's sister, and new branch lengths. The tip ID must only contain letters numbers and underscores. Optionally, user can specify the IDs for the new parental internal nodes. Ensure that the strt_age is greater than the end_age, and that the strt_age falls within the age span of the sister ID. Otherwise, negative spns may be produced leading to an error. Note, returned tree will not have a node matrix. Note, providing negative end ages will increase the age of the tree.

See Also

rmTips, https://github.com/DomBennett/treeman/wiki/manip-methods

Examples

```
library(treeman)
tree <- randTree(10)
tree_age <- getAge(tree)
possible_ages <- getSpnAge(tree, 't1', tree_age)
start_age <- runif(1, possible_ages[['end']], possible_ages[['start']])
end_age <- possible_ages[['end']]
tree <- addTip(tree, tid='t11', sid='t1', strt_age=start_age,
end_age=end_age, tree_age=tree_age)
summary(tree)
```

---

birds *Phylogenetic tree of Aves*

Description

Jetz et al. (2012)'s Avian supertree augmented with taxonomic information generated from the NCBI taxonomy. Here used for testing and demonstrating treeman functions. See R script to see how the tree was generate: https://github.com/DomBennett/treeman/blob/master/other/generate_tree_data.R.

Usage

```
data(birds)
```

Format

`birds` is a `TreeMan` object

Source

Jetz, W., Thomas, G.H., Joy, J.B., Hartmann, K. and Mooers, A.O. 2012. The global diversity of birds in space and time. Nature, 491: 444-448

Examples

```
data(birds) # load object
summary(birds)
```

blncdTree *Generate a balanced tree*

Description

Returns a balanced TreeMan tree with n tips.

Usage

```
blncdTree(n, wndmtrx = FALSE, parallel = FALSE)
```

Arguments

n	number of tips, integer, must be 3 or greater
wndmtrx	T/F add node matrix? Default FALSE.
parallel	T/F run in parallel? Default FALSE.

Details

Equivalent to ape's stree(type='balanced') but returns a TreeMan tree. Tree is always rooted and bifurcating.

See Also

TreeMan-class, randTree, unblncdTree

Examples

```
library(treeman)
tree <- blncdTree(5)
```

calcDstBLD *Calculate the BLD between two trees*

Description

Returns the branch length distance between two trees.

Usage

```
calcDstBLD(tree_1, tree_2, nrmlsd = FALSE, parallel = FALSE,
  progress = "none")
```

Arguments

tree_1	TreeMan object
tree_2	TreeMan object
nrmlsd	Boolean, should returned value be between 0 and 1? Default, FALSE.
parallel	logical, make parallel?
progress	name of the progress bar to use, see create_progress_bar

Details

BLD is the Robinson-Foulds distance weighted by branch length. Instead of summing the differences in partitions between the two trees, the metric takes the square root of the squared difference in branch lengths. Parallelizable.

References

Kuhner, M. K. and Felsenstein, J. (1994) Simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates. Molecular Biology and Evolution, 11, 459-468.

See Also

calcDstTrp, calcDstRF https://github.com/DomBennett/treeman/wiki/calc-methods

Examples

```
library(treeman)
tree_1 <- randTree(10)
tree_2 <- randTree(10)
calcDstBLD(tree_1, tree_2)
```

---

calcDstMtrx *Calculate the distance matrix*

Description

Returns a distance matrix for specified ids of a tree.

Usage

```
calcDstMtrx(tree, ids, parallel = FALSE, progress = "none")
```

Arguments

tree	TreeMan object
ids	IDs of nodes/tips
parallel	logical, make parallel?
progress	name of the progress bar to use, see create_progress_bar

Details

The distance between every id in the tree is calculated by summing the lengths of the branches that connect them. This can be useful for testing the distances between trees, checking for evoltuionary isolated tips etc. Parallelizable.

See Also

calcDstBLD, calcDstRF, calcDstTrp https://github.com/DomBennett/treeman/wiki/calc-methods

Examples

```
# checking the distance between two trees
library(treeman)
tree_1 <- randTree(10)
tree_2 <- randTree(10)
dmat1 <- calcDstMtrx(tree_1, tree_1['tips'])
dmat2 <- calcDstMtrx(tree_2, tree_2['tips'])
mdl <- cor.test(x=dmat1, y=dmat2)
as.numeric(1 - mdl$estimate)  # 1 - Pearson's r
```

---

calcDstRF *Calculate the Robinson-Foulds distance between two trees*

Description

Returns the Robinson-Foulds distance between two trees.

Usage

```
calcDstRF(tree_1, tree_2, nrmlsd = FALSE, parallel = FALSE,
  progress = "none")
```

Arguments

tree_1	TreeMan object
tree_2	TreeMan object
nrmlsd	Boolean, should returned value be between 0 and 1? Default, FALSE.
parallel	logical, make parallel?
progress	name of the progress bar to use, see create_progress_bar

Details

RF distance is calculated as the sum of partitions in one tree that are not shared by the other. The maximum number of split differences is the total number of nodes in both trees (excluding the roots). Trees are assumed to be bifurcating, this is not tested. The metric is calculated as if trees are unrooted. Parallelizable.

References

Robinson, D. R.; Foulds, L. R. (1981). "Comparison of phylogenetic trees". Mathematical Biosciences 53: 131-147.

See Also

calcDstBLD, calcDstTrp https://github.com/DomBennett/treeman/wiki/calc-methods

Examples

```
library(treeman)
tree_1 <- randTree(10)
tree_2 <- randTree(10)
calcDstRF(tree_1, tree_2)
```

---

calcDstTrp *Calculate the triplet distance between two trees*

Description

Returns the triplet distance between two trees.

Usage

```
calcDstTrp(tree_1, tree_2, nrmlsd = FALSE, parallel = FALSE,
  progress = "none")
```

Arguments

tree_1	TreeMan object
tree_2	TreeMan object
nrmlsd	Boolean, should returned value be between 0 and 1? Default, FALSE.
parallel	logical, make parallel?
progress	name of the progress bar to use, see create_progress_bar

Details

The triplet distance is calculated as the sum of different outgroups among every triplet of tips between the two trees. Normalisation is performed by dividing the resulting number by the total number of triplets shared between the two trees. The triplet distance is calculated only for shared tips between the two trees. Parallelizable.

References

Critchlow DE, Pearl DK, Qian C. (1996) The Triples Distance for rooted bifurcating phylogenetic trees. Systematic Biology, 45, 323-34.

See Also

calcDstBLD, calcDstRF https://github.com/DomBennett/treeman/wiki/calc-methods

Examples

```
library(treeman)
tree_1 <- randTree(10)
tree_2 <- randTree(10)
calcDstTrp(tree_1, tree_2)
```

---

calcFrPrp *Calculate evolutionary distinctness*

Description

Returns the evolutationary distinctness of ids using the fair proportion metric.

Usage

```
calcFrPrp(tree, tids, progress = "none")
```

Arguments

tree	TreeMan object
tids	tip IDs
progress	name of the progress bar to use, see create_progress_bar

Details

The fair proportion metric calculates the evolutionary distinctness of tips in a tree through summing the total amount of branch length each tip represents, where each branch in the tree is evenly divided between all descendants. Parallelizable.

References

Isaac, N.J.B., Turvey, S.T., Collen, B., Waterman, C. and Baillie, J.E.M. (2007). Mammals on the EDGE: conservation priorities based on threat and phylogeny. PLoS ONE, 2, e296.

See Also

calcPhyDv, calcPrtFrPrp, https://github.com/DomBennett/treeman/wiki/calc-methods

Examples

```
library(treeman)
tree <- randTree(10)
calcFrPrp(tree, tree['tips'])
```

calcNdBlnc *Calculate the balance of a node*

Description

Returns the balance of a node.

Usage

```
calcNdBlnc(tree, id)
```

Arguments

tree	TreeMan object
id	node id

Details

Balance is calculated as the absolute difference between the number of descendents of the two bifurcating edges of a node and the expected value for a balanced tree. NA is returned if the node is polytomous or a tip.

See Also

calcNdsBlnc, https://github.com/DomBennett/treeman/wiki/calc-methods

Examples

```
library(treeman)
tree <- randTree(10)
calcNdBlnc(tree, id=tree['root'])  # root balance
```

calcNdsBlnc *Calculate the balances of all nodes*

Description

Returns the absolute differences in number of descendants for bifurcating branches of every node

Usage

```
calcNdsBlnc(tree, ids, parallel = FALSE, progress = "none")
```

Arguments

tree	TreeMan object
ids	node ids
parallel	logical, make parallel?
progress	name of the progress bar to use, see create_progress_bar

Details

Runs calcNdBlnc() across all node IDs. NA is returned if the node is polytomous. Parallelizable.

See Also

calcNdBlnc, https://github.com/DomBennett/treeman/wiki/calc-methods

Examples

```
library(treeman)
tree <- randTree(10)
calcNdsBlnc(tree, ids=tree['nds'])
```

---

calcOvrlp *Calculate phylogenetic overlap*

Description

Returns the sum of branch lengths represented by ids_1 and ids_2 for a tree.

Usage

```
calcOvrlp(tree, ids_1, ids_2, nrmlsd = FALSE, parallel = FALSE,
  progress = "none")
```

Arguments

tree	TreeMan object
ids_1	tip ids of community 1
ids_2	tip ids of community 2
nrmlsd	Boolean, should returned value be between 0 and 1? Default, FALSE.
parallel	logical, make parallel?
progress	name of the progress bar to use, see create_progress_bar

Details

Use this to calculate the sum of branch lengths that are represented between two communities. This measure is also known as the unique fraction. It can be used to measure concepts of phylogenetic turnover. Parallelizable.

References

Lozupone, C., & Knight, R. (2005). UniFrac: a new phylogenetic method for comparing microbial communities. Applied and Environmental Microbiology, 71(12), 8228-35.

See Also

calcPhyDv https://github.com/DomBennett/treeman/wiki/calc-methods

Examples

```
library(treeman)
tree <- randTree(10)
ids_1 <- sample(tree['tips'], 5)
ids_2 <- sample(tree['tips'], 5)
calcOvrlp(tree, ids_1, ids_2)
```

---

calcPhyDv *Calculate phylogenetic diversity*

Description

Returns the phylogenetic diversity of a tree for the tips specified.

Usage

```
calcPhyDv(tree, tids, parallel = FALSE, progress = "none")
```

Arguments

tree	TreeMan object
tids	tip ids
parallel	logical, make parallel?
progress	name of the progress bar to use, see create_progress_bar

Details

Faith's phylogenetic diversity is calculated as the sum of all connected branches for specified tips in a tree. It can be used to investigate how biodviersity as measured by the phylogeny changes. Parallelizable. The function uses getCnntdNds().

References

Faith, D. (1992). Conservation evaluation and phylogenetic diversity. Biological Conservation, 61, 1-10.

See Also

calcFrPrp, calcOvrlp, getCnnctdNds, https://github.com/DomBennett/treeman/wiki/calc-methods

Examples

```
library(treeman)
tree <- randTree(10)
calcPhyDv(tree, tree['tips'])
```

`calcPrtFrPrp` *Calculate evolutionary distinctness for part of tree*

Description

Returns the evolutationary distinctness of ids using the fair proportion metric.

Usage

```
calcPrtFrPrp(tree, tids, ignr = NULL, progress = "none")
```

Arguments

`tree`	`TreeMan` object
`tids`	tip IDs
`ignr`	tips to ignore in calculation
`progress`	name of the progress bar to use, see `create_progress_bar`

Details

Extension of `calcFrPrp()` but with ignore argument. Use `ignr` to ignore certain tips from calculation. For example, if any of tips are extinct you may wish to ignore these.

References

Isaac, N.J.B., Turvey, S.T., Collen, B., Waterman, C. and Baillie, J.E.M. (2007). Mammals on the EDGE: conservation priorities based on threat and phylogeny. PLoS ONE, 2, e296.

See Also

`calcFrPrp` https://github.com/DomBennett/treeman/wiki/calc-methods

Examples

```
library(treeman)
tree <- randTree(10)
calcPrtFrPrp(tree, c('t1','t3'), ignr='t2')
```

checkNdlst *Check if ndlst is correct*

Description

Return T/F fpr `ndlst` consistency

Usage

```
checkNdlst(ndlst, root)
```

Arguments

`ndlst`	`ndlst`
`root`	root ID

Details

Tests whether each node in tree points to valid other node IDs. Also ensures ‘spn‘ and ‘root‘ are correct. Reports nodes that have errors.

See Also

`fastCheckTreeMan`, `checkTreeMen`

Examples

```
library(treeman)
tree <- randTree(100)
(checkNdlst(tree@ndlst, tree@root))
```

checkTreeMen *Check if trees are correct*

Description

Return T/F if trees is a true `TreeMen` object

Usage

```
checkTreeMen(object)
```

Arguments

`object`	`TreeMen` object

Details

Tests whether all trees in object are TreeMan objects

See Also

checkNdlst

---

cTrees *cTrees*

Description

Return TreeMen of concatenated trees.

Usage

```
cTrees(x, ...)
```

Arguments

x	TreeMan or TreeMen objects
...	more TreeMan or TreeMen objects

Details

Concatenate trees into single TreeMen object.

See Also

TreeMen-class, TreeMan-class, list-to-TreeMen

Examples

```
library(treeman)
trees <- cTrees(randTree(10), randTree(10))
```

fastCheckTreeMan *Check if tree is correct, fast!*

Description

Return T/F if tree is a true TreeMan object

Usage

```
fastCheckTreeMan(object)
```

Arguments

object	TreeMan object

Details

Whenever a tree is first initiated this check is used. For more detailed checking use checkNdlst.

See Also

checkNdlst, checkTreeMen

getAge *Get age of tree*

Description

Returns age, numeric, of tree

Usage

```
getAge(tree, parallel = FALSE)
```

Arguments

tree	TreeMan object
parallel	logical, make parallel?

Details

Calculates the age of a tree, determined as the maximum tip to root distance.

See Also

updateSlts, https://github.com/DomBennett/treeman/wiki/get-methods

Examples

```
library(treeman)
tree <- randTree(10)
(getAge(tree))
```

---

getCnnctdNds *Get all nodes connected by given tips*

Description

Return a vector of IDs of all nodes that are connected to tip IDs given.

Usage

```
getCnnctdNds(tree, tids)
```

Arguments

tree	TreeMan object
tids	vector of tip IDs

Details

Returns a vector. This function is the basis for `calcPhyDv()`, it determines the unique set of nodes connected for a set of tips.

See Also

getUnqNds, calcFrPrp, calcPhyDv

Examples

```
library(treeman)
tree <- randTree(10)
cnntdnds <- getCnnctdNds(tree, c('t1', 't2'))
```

getDcsd *Get extinct tips from a tree*

Description

Return all extinct tip IDs.

Usage

```
getDcsd(tree, tol = 1e-08)
```

Arguments

tree	TreeMan object
tol	zero tolerance

Details

Returns a vector.

See Also

getLvng, isUltrmtrc, https://github.com/DomBennett/treeman/wiki/get-methods

Examples

```
library(treeman)
tree <- randTree(10)
(getDcsd(tree))
```

getLvng *Get extant tips from a tree*

Description

Return all extant tip IDs.

Usage

```
getLvng(tree, tol = 1e-08)
```

Arguments

tree	TreeMan object
tol	zero tolerance

Details

Returns a vector.

See Also

getDcsd, isUltrmtrc, https://github.com/DomBennett/treeman/wiki/get-methods

Examples

```
library(treeman)
tree <- randTree(10)
(getLvng(tree))
```

---

getNdAge *Get age*

Description

Return the age for id. Requires the known age of the tree to be provided.

Usage

```
getNdAge(tree, id, tree_age)
```

Arguments

tree	TreeMan object
id	node id
tree_age	numeric value of known age of tree

Details

Returns a numeric.

See Also

getNdsAge, getSpnAge, getSpnsAge, getPrnt, getAge https://github.com/DomBennett/treeman/wiki/get-methods

Examples

```
library(treeman)
data(mammals)
# when did apes emerge?
# get parent id for all apes
prnt_id <- getPrnt(mammals, ids=c('Homo_sapiens', 'Hylobates_concolor'))
# mammal_age <- getAge(mammals)  # ~166.2, needs to be performed when tree is not up-to-date
getNdAge(mammals, id=prnt_id, tree_age=166.2)
```

getNdKids *Get children IDs*

Description

Return the node ids of all tips that descend from node.

Usage

```
getNdKids(tree, id)
```

Arguments

tree	TreeMan object
id	node id

Details

Returns a vector

See Also

getNdsKids, https://github.com/DomBennett/treeman/wiki/get-methods

Examples

```
library(treeman)
tree <- randTree(10)
# everyone descends from root
getNdKids(tree, id=tree['root'])
```

getNdLng *Get lineage*

Description

Return unique taxonomic names for connecting id to root.

Usage

```
getNdLng(tree, id)
```

Arguments

tree	TreeMan object
id	node id

Details

Returns a vector.

See Also

getNdsLng, getNdsFrmTxnyms, https://github.com/DomBennett/treeman/wiki/get-methods

Examples

```
library(treeman)
data(mammals)
# return human lineage
getNdLng(mammals, id='Homo_sapiens')
```

---

getNdPD *Get phylogenetic diversity of node*

Description

Return summed value of all descending spns

Usage

```
getNdPD(tree, id)
```

Arguments

tree	TreeMan object
id	node id

Details

Sums the lengths of all descending branches from a node.

See Also

getNdsPD, https://github.com/DomBennett/treeman/wiki/get-methods

Examples

```
library(treeman)
tree <- randTree(10)
getNdPD(tree, id='n1')  # return PD of n1 which in this case is for the whole tree
```

getNdPrdst *Get pre-distance*

Description

Return root to tip distance (prdst) for `id`

Usage

```
getNdPrdst(tree, id)
```

Arguments

`tree`	`TreeMan` object
`id`	node id

Details

Sums the lengths of all branches from `id` to root.

See Also

`getNdsPrdst`, https://github.com/DomBennett/treeman/wiki/get-methods

Examples

```
library(treeman)
tree <- randTree(10)
getNdPrdst(tree, id='t1')  # return the distance to root from t1
```

getNdPrids *Get pre-nodes to root*

Description

Return node ids for connecting `id` to root.

Usage

```
getNdPrids(tree, id)
```

Arguments

`tree`	`TreeMan` object
`id`	node id

Details

Returns a vector. IDs are returned order from node ID to root.

See Also

getNdsPrids, getNdPtids, getNdsPtids, https://github.com/DomBennett/treeman/wiki/get-methods

Examples

```
library(treeman)
tree <- randTree(10)
# get all nodes to root
getNdPrids(tree, id='t1')
```

---

getNdPtids *Get post-nodes to tips*

Description

Return node ids for connecting `id` to kids.

Usage

```
getNdPtids(tree, id)
```

Arguments

`tree`	TreeMan object
`id`	node id

Details

Returns a vector.

See Also

getNdsPtids, getNdPrids, getNdsPrids, https://github.com/DomBennett/treeman/wiki/get-methods

Examples

```
library(treeman)
tree <- randTree(10)
# get all nodes from root to tip
getNdPtids(tree, id='n1')
```

getNdsAge *Get ages for multiple nodes*

Description

Return the age for `ids`.

Usage

```
getNdsAge(tree, ids, tree_age, parallel = FALSE, progress = "none")
```

Arguments

`tree`	`TreeMan` object
`ids`	vector of node ids
`tree_age`	numeric value of known age of tree
`parallel`	logical, make parallel?
`progress`	name of the progress bar to use, see `create_progress_bar`

Details

Returns a vector, parallelizable.

See Also

`getNdAge`, `getSpnAge`, `getSpnsAge`, `https://github.com/DomBennett/treeman/wiki/get-methods`

Examples

```
library(treeman)
tree <- randTree(10)
getNdsAge(tree, ids=tree['nds'], tree_age=getAge(tree))
```

getNdsFrmTxnyms *Get IDs for nodes represented txnyms*

Description

Return a list of IDs for any node that contains the given txnyms.

Usage

```
getNdsFrmTxnyms(tree, txnyms)
```

Arguments

`tree`	`TreeMan` object
`txnyms`	vector of taxonomic group names

Details

Returns a list. Txnyms must be spelt correctly.

See Also

`taxaResolve`, `setTxnyms`, `searchTxnyms`, `getNdsLng`, `getNdLng`

Examples

```
library(treeman)
data(mammals)
# what ID represents the apes?
getNdsFrmTxnyms(mammals, 'Hominoidea')
```

`getNdsKids` *Get children IDs for multiple nodes*

Description

Return the node ids of all tips that descend from each node in `ids`.

Usage

```
getNdsKids(tree, ids, parallel = FALSE, progress = "none")
```

Arguments

`tree`	`TreeMan` object
`ids`	vector of node ids
`parallel`	logical, make parallel?
`progress`	name of the progress bar to use, see `create_progress_bar`

Details

Returns a list, parallelizable.

See Also

`getNdKids`, https://github.com/DomBennett/treeman/wiki/get-methods

Examples

```
library(treeman)
tree <- randTree(10)
getNdsKids(tree, id=tree['nds'])
```

---

getNdsLng *Get lineage for multiple nodes*

Description

Return unique taxonyms for connecting `ids` to root.

Usage

```
getNdsLng(tree, ids, parallel = FALSE, progress = "none")
```

Arguments

`tree`	`TreeMan` object
`ids`	vector of node ids
`parallel`	logical, make parallel?
`progress`	name of the progress bar to use, see `create_progress_bar`

Details

Returns a list, parallelizable.

See Also

`getNdLng`, `getNdsFrmTxnyms`, https://github.com/DomBennett/treeman/wiki/get-methods

Examples

```
library(treeman)
data(mammals)
# return human and gorilla lineages
getNdsLng(mammals, id=c('Homo_sapiens', 'Gorilla_gorilla'))
```

getNdSlt *Get a node slot*

Description

Returns the value of named slot.

Usage

```
getNdSlt(tree, slt_nm, id)
```

Arguments

tree	TreeMan object
slt_nm	slot name
id	node id

Details

Returned object depends on name, either character, vector or numeric. Default node slots are: id, spn, prid, ptid and txnym. If slot is empty, returns NA.

See Also

getNdsSlt, https://github.com/DomBennett/treeman/wiki/get-methods

Examples

```
library(treeman)
tree <- randTree(10)
getNdSlt(tree, slt_nm='spn', id='t1')  # return span of t1
```

getNdsPD *Get phylogenetic diversities of nodes*

Description

Return summed value of all descending spns

Usage

```
getNdsPD(tree, ids, parallel = FALSE, progress = "none")
```

Arguments

tree	TreeMan object
ids	vector of node ids
parallel	logical, make parallel?
progress	name of the progress bar to use, see create_progress_bar

Details

Sums the lengths of all descending branches from a node.

See Also

getNdPD, https://github.com/DomBennett/treeman/wiki/get-methods

Examples

```
library(treeman)
tree <- randTree(10)
getNdsPD(tree, ids=tree['all'])  # return PD of all ids
```

---

getNdsPrdst *Get pre-distances*

Description

Return root to tip distances (prdst) for ids

Usage

```
getNdsPrdst(tree, ids, parallel = FALSE, progress = "none")
```

Arguments

tree	TreeMan object
ids	vector of node ids
parallel	logical, make parallel?
progress	name of the progress bar to use, see create_progress_bar

Details

Sums the lengths of all branches from ids to root.

See Also

getNdPrdst, https://github.com/DomBennett/treeman/wiki/get-methods

Examples

```
library(treeman)
tree <- randTree(10)
getNdsPrdst(tree, ids=tree['tips'])  # return prdsts for all tips
```

`getNdsPrids` *Get pre-nodes for multiple nodes*

Description

Return node ids for connecting `id` to root.

Usage

```
getNdsPrids(tree, ids, ordrd = FALSE, parallel = FALSE, progress = "none")
```

Arguments

`tree`	`TreeMan` object
`ids`	vector of node ids
`ordrd`	logical, ensure returned prids are ordered ID to root
`parallel`	logical, make parallel?
`progress`	name of the progress bar to use, see `create_progress_bar`

Details

Returns a list, parallizable. The function will work faster if `ordrd` is FALSE.

See Also

`getNdPrids`, `getNdPtids`, `getNdsPtids`, https://github.com/DomBennett/treeman/wiki/get-methods

Examples

```
library(treeman)
tree <- randTree(10)
getNdsPrids(tree, ids=tree['tips'])
```

getNdsPtids *Get post-nodes to tips for multiple nodes*

Description

Return node ids for connecting `ids` to kids.

Usage

```
getNdsPtids(tree, ids, parallel = FALSE, progress = "none")
```

Arguments

`tree`	`TreeMan` object
`ids`	vector of node ids
`parallel`	logical, make parallel?
`progress`	name of the progress bar to use, see `create_progress_bar`

Details

Returns a list, parallizable.

See Also

`getNdPtids`, `getNdPrids`, `getNdsPrids`, `https://github.com/DomBennett/treeman/wiki/get-methods`

Examples

```
library(treeman)
tree <- randTree(10)
# get all nodes to tip for all nodes
getNdsPtids(tree, ids=tree['nds'])
```

getNdsSlt *Get a node slot for multiple nodes*

Description

Returns the values of named slot as a vector for atomic values, else list.

Usage

```
getNdsSlt(tree, slt_nm, ids, parallel = FALSE, progress = "none")
```

Arguments

tree	TreeMan object
slt_nm	slot name
ids	vector of node ids
parallel	logical, make parallel?
progress	name of the progress bar to use, see create_progress_bar

Details

Returned object depends on name, either character, vector or numeric. Parallelizable. Default node slots are: id, spn, prid, ptid and txnym.

See Also

getNdSlt, https://github.com/DomBennett/treeman/wiki/get-methods

Examples

```
library(treeman)
tree <- randTree(10)
getNdsSlt(tree, slt_nm='spn', ids=tree['tips'])  # return spans of all tips
```

---

getNdsSstr *Get sister id*

Description

Returns the ids of the sister(s) of nd ids given.

Usage

```
getNdsSstr(tree, ids, parallel = FALSE, progress = "none")
```

Arguments

tree	TreeMan object
ids	nd ids
parallel	logical, make parallel?
progress	name of the progress bar to use, see create_progress_bar

Details

An error is raised if there is no sister (e.g. for the root). There can be more than one sister if tree is polytomous. Parallelizable.

See Also

getNdSstr, https://github.com/DomBennett/treeman/wiki/get-methods

Examples

```
library(treeman)
tree <- randTree(10)
getNdsSstr(tree, ids=tree['tips'])
```

---

getNdSstr *Get sister id*

Description

Returns the id of the sister(s) of node id given.

Usage

```
getNdSstr(tree, id)
```

Arguments

tree	TreeMan object
id	node id

Details

An error is raised if there is no sister (e.g. for the root). There can be more than one sister if tree is polytomous.

See Also

getNdsSstr, https://github.com/DomBennett/treeman/wiki/get-methods

Examples

```
library(treeman)
tree <- randTree(10)
getNdSstr(tree, id='t1')
```

getOtgrp *Get outgroup*

Description

Return the outgroup based on a tree and a vector of IDs.

Usage

```
getOtgrp(tree, ids)
```

Arguments

tree	TreeMan object
ids	vector of node ids

Details

Returns a id, character. If there are multiple possible outgroups, returns NULL.

See Also

https://github.com/DomBennett/treeman/wiki/get-methods

Examples

```
library(treeman)
data(mammals)
# orangutan is an outgroup wrt humans and chimps
getOtgrp(mammals, ids=c('Homo_sapiens', 'Pan_troglodytes', 'Pongo_pygmaeus'))
```

getPath *Get path between nodes*

Description

Return node ids for connecting from to to.

Usage

```
getPath(tree, from, to)
```

Arguments

tree	TreeMan object
from	starting node id
to	ending node id

Details

Returns a vector, first id is `from` to `to`.

See Also

https://github.com/DomBennett/treeman/wiki/get-methods

Examples

```
library(treeman)
data(mammals)
# what's the phylogenetic distance from humans to gorillas?
ape_id <- getPrnt(mammals, ids=c('Homo_sapiens', 'Hylobates_concolor'))
pth <- getPath(mammals, from='Homo_sapiens', to='Gorilla_gorilla')
sum(getNdsSlt(mammals, ids=pth, slt_nm='spn'))
```

`getPrnt` *Get parent*

Description

Return parental (most recent common ancestor) node id for `ids`.

Usage

```
getPrnt(tree, ids)
```

Arguments

`tree`	`TreeMan` object
`ids`	vector of node ids

Details

Returns a character.

See Also

`getSubtree`, https://github.com/DomBennett/treeman/wiki/get-methods

Examples

```
library(treeman)
data(mammals)
# choosing ids from the two main branches of apes allows to find the parent for all apes
ape_id <- getPrnt(mammals, ids=c('Homo_sapiens', 'Hylobates_concolor'))
```

getSpnAge *Get age range*

Description

Return start and end ages for `id` from when it first appears to when it splits

Usage

```
getSpnAge(tree, id, tree_age)
```

Arguments

`tree`	`TreeMan` object
`id`	node id
`tree_age`	numeric value of known age of tree

Details

Returns a dataframe.

See Also

`getNdAge`, `getNdsAge`, `getSpnsAge`, https://github.com/DomBennett/treeman/wiki/get-methods

Examples

```
library(treeman)
data(mammals)
# mammal_age <- getAge(mammals)  # ~166.2, needs to be performed when tree is not up-to-date
getSpnAge(mammals, id='Homo_sapiens', tree_age=166.2)
```

getSpnsAge *Get age ranges for multiple nodes*

Description

Return start and end ages for `ids` from when they first appear to when they split

Usage

```
getSpnsAge(tree, ids, tree_age, parallel = FALSE, progress = "none")
```

Arguments

tree	TreeMan object
ids	vector of node ids
tree_age	numeric value of known age of tree
parallel	logical, make parallel?
progress	name of the progress bar to use, see create_progress_bar

Details

Returns a dataframe, parallelizable.

See Also

getNdAge, getNdsAge, getSpnAge, https://github.com/DomBennett/treeman/wiki/get-methods

Examples

```
library(treeman)
tree <- randTree(10)
# all nodes but root
ids <- tree['nds'][tree['nds'] != tree['root']]
getSpnsAge(tree, ids=ids, tree_age=getAge(tree))
```

---

getSubtree *Get subtree*

Description

Return tree descending from id.

Usage

```
getSubtree(tree, id)
```

Arguments

tree	TreeMan object
id	node id

Details

Returns a TreeMan, parallelizable. id must be an internal node.

See Also

getPrnt, addClade, https://github.com/DomBennett/treeman/wiki/get-methods

Examples

```
library(treeman)
data(mammals)
# get tree of apes
ape_id <- getPrnt(mammals, ids=c('Homo_sapiens', 'Hylobates_concolor'))
apes <- getSubtree(mammals, id=ape_id)
summary(apes)
```

---

getUnqNds *Get unique nodes represented by tips*

Description

Return a list of IDs for any node that are represented by tip IDs given.

Usage

```
getUnqNds(tree, tids)
```

Arguments

tree	TreeMan object
tids	vector of tip IDs

Details

Returns a vector.

See Also

getCnnctdNds, calcFrPrp, calcPhyDv

Examples

```
library(treeman)
tree <- randTree(10)
unqnds <- getUnqNds(tree, c('t1', 't2'))
```

isUltrmtrc *Is tree ultrametric?*

Description

Return TRUE if all tips end at 0, else FALSE.

Usage

```
isUltrmtrc(tree, tol = 1e-08)
```

Arguments

tree	TreeMan object
tol	zero tolerance

Details

Returns a boolean. This function works in the background for the ['ultr'] slot in a TreeMan object.

See Also

getLvng, getDcsd

Examples

```
library(treeman)
tree <- randTree(10)
(isUltrmtrc(tree))
```

list-to-TreeMen *Convert list to a TreeMen*

Description

Return a TreeMen object from a list of TreeMans

See Also

TreeMen-class

Examples

```
library(treeman)
trees <- list('tree_1'=randTree(10), 'tree_2'=randTree(10))
trees <- as(trees, 'TreeMen')
```

loadTreeMan *Load a TreeMan object in serialization format*

Description

TreeMan equivalent to load() but able to handle node matrices.

Usage

```
loadTreeMan(file)
```

Arguments

file	file path

Details

It is not possible to use save() on TreeMan objects with node matrices. Node matrices are bigmemory matrices and are therefore outside the R environment, see bigmemory documentation for more information. Saving and loading a bigmemory matrix may cause memory issues in R and cause R to crash.

This function can safely read a TreeMan object with and without a node matrix. saveTreeMan() function stores the tree using the serialization format and the node matrix as a hidden .csv. Both parts of the tree can be reloaded to an R environment with loadTreeMan(). The hidden node matrix filename is based on the file argument: file + _ndmtrx

Reading and writing trees with saveTreeMan() and loadTreeMan is faster than any of the other read and write functions.

See Also

saveTreeMan, readTree,writeTree, readTrmn, writeTrmn

Examples

```
library(treeman)
tree <- randTree(100, wndmtrx=TRUE)
saveTreeMan(tree, file='test.RData')
rm(tree)
tree <- loadTreeMan(file='test.RData')
file.remove('test.RData', 'testRData_ndmtrx')
```

mammals *Phylogenetic tree of Mammalia*

Description

Bininda-Emonds et al. (2007)'s Mammalian supertree augmented with taxonomic information. Here used for testing and demonstrating treeman functions. See R script to see how the tree was generate: https://github.com/DomBennett/treeman/blob/master/other/generate_tree_data.R.

Usage

```
data(mammals)
```

Format

`mammals` is a `TreeMan` object

Source

Bininda-Emonds et al. 2007. The Delayed Rise of Present-Day Mammals. Nature, 446(7135): 507-512

Examples

```
data(mammals) # load object
summary(mammals)
summary(mammals[['Homo_sapiens']])
```

multiPhylo-class *multiPhylo class*

Description

multiPhylo class

multiPhylo-to-TreeMen *Convert multiPhylo to TreeMen*

Description

Return a TreeMen from ape's mutlPhylo

See Also

TreeMan-to-phylo, phylo-to-TreeMan, TreeMen-to-multiPhylo TreeMan-class

Examples

```
library(treeman)
library(ape)
trees <- c(rtree(10), rtree(10), rtree(10))
trees <- as(trees, 'TreeMen')
```

Node-class *Node-class*

Description

The Node is an S4 class used for displaying node information. It is only generated when a user implements the [[]] on a tree. Information is only accurate if tree has been updated with updateTree().

Usage

```
## S4 method for signature 'Node'
as.character(x)

## S4 method for signature 'Node'
show(object)

## S4 method for signature 'Node'
print(x)

## S4 method for signature 'Node'
summary(object)

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

Arguments

x	Node object
object	Node object
i	slot name
j	missing
...	missing
drop	missing

Slots

id unique ID for node in tree['ndlst']

spn length of preceding branch

prid parent node ID

ptid child node ID

kids descending tip IDs

nkids number of descending tip IDs

txnym list of associated taxonyms

pd total branch length represented by node

prdst total branch length of connected prids

root T/F root node?

tip T/F tip node?

See Also

TreeMan-class, TreeMen-class

---

phylo-class *phylo class*

Description

phylo class

phylo-to-TreeMan *Convert phylo to TreeMan*

Description

Return a TreeMan from ape's phylo

See Also

TreeMan-to-phylo, TreeMen-to-multiPhylo multiPhylo-to-TreeMen TreeMan-class

Examples

```
library(treeman)
library(ape)
tree <- compute.brlen(rtree(10))
tree <- as(tree, 'TreeMan')
```

pinTips *Pin tips to a tree*

Description

Returns a tree with new tips added based on given lineages and time points

Usage

```
pinTips(tree, tids, lngs, end_ages, tree_age)
```

Arguments

tree	TreeMan object
tids	new tip ids
lngs	list of vectors of the lineages of each tid (ordered high to low rank)
end_ages	end time points for each tid
tree_age	age of tree

Details

User must provide a vector of new tip IDs, a list of the ranked lineages for these IDs (in ascending order) and a vector of end time points for each new ID (0s for extant tips). The function expects the given tree to be taxonomically informed; the txnym slot for every node should have a taxonomic label. The function takes the lineage and tries to randomly add the new tip at the lowest point in the taxonomic rank before the end time point. Note, returned tree will not have a node matrix.

See Also

addTip, rmTips, https://github.com/DomBennett/treeman/wiki/manip-methods

Examples

```
# see https://github.com/DomBennett/treeman/wiki/Pinning-tips for a detailed example
```

---

plants *Phylogenetic tree of European Flora*

Description

Durka and Michalski (2012)'s large phylogenetic tree of European vascular plants augmented with taxonomic information generated from the NCBI taxonomy. Here used for testing and demonstrating treeman functions. See R script to see how the tree was generate: https://github.com/DomBennett/treeman/blob/master/other/generate_tree_data.R.

Usage

```
data(plants)
```

Format

plants is a TreeMan object

Source

Durka, W. and Michalski, S.G. 2012. Daphne: a dated phylogeny of a large European flora for phylogenetically informed ecological analyses. Ecology 93:2297-2297.

Examples

```
data(plants) # load object
summary(plants)
```

pstMnp *Update prinds and tinds*

Description

Return tree with updated slots.

Usage

```
pstMnp(tree)
```

Arguments

tree	TreeMan object

Details

This function is automatically run. Only run, if you are creating yor own functions to add and remove elements of the ndlst.

See Also

updateSlts, addNdmtrx, getAge

randTree *Generate a random tree*

Description

Returns a random TreeMan tree with n tips.

Usage

```
randTree(n, wndmtrx = FALSE, parallel = FALSE)
```

Arguments

n	number of tips, integer, must be 3 or greater
wndmtrx	T/F add node matrix? Default FALSE.
parallel	T/F run in parallel? Default FALSE.

Details

Equivalent to ape's rtree() but returns a TreeMan tree. Tree is always rooted and bifurcating.

See Also

TreeMan-class, blncdTree, unblncdTree

Examples

```
library(treeman)
tree <- randTree(5)
```

---

readTree *Read a Newick tree*

Description

Return a TreeMan or TreeMen object from a Newick treefile

Usage

```
readTree(file = NULL, text = NULL, spcl_slt_nm = "Unknown",
  wndmtrx = FALSE, parallel = FALSE, progress = "none")
```

Arguments

file	file path
text	Newick character string
spcl_slt_nm	name of special slot for internal node labels, default 'Unknown'.
wndmtrx	T/F add node matrix? Default FALSE.
parallel	logical, make parallel?
progress	name of the progress bar to use, see create_progress_bar

Details

Read a single or multiple trees from a file, or a text string. Parallelizable when reading multiple trees. The function will add any internal node labels in the Newick tree as a user-defined data slots. The name of this slot is defined with the spcl_slt_nm. These data can be accessed/manipulated with the `getNdsSlt()` function. Trees are always read as rooted. (Unrooted trees have polytomous root nodes.)

See Also

https://en.wikipedia.org/wiki/Newick_format, addNdmtrx, writeTree, randTree, readTrmn, writeTrmn, saveTreeMan, loadTreeMan

Examples

```
library(treeman)
# tree string with internal node labels as bootstrap results
tree <- readTree(text="((A:1.0,B:1.0)0.9:1.0,(C:1.0,D:1.0)0.8:1.0)0.7:1.0;",
spcl_slt_nm='bootstrap')
# retrieve bootstrap values by node
tree['bootstrap']
```

---

readTrmn *Read a .trmn tree*

Description

Return a TreeMan or TreeMen object from a .trmn treefile

Usage

```
readTrmn(file, wndmtrx = FALSE, parallel = FALSE, progress = "none")
```

Arguments

file	file path
wndmtrx	T/F add node matrix? Default FALSE.
parallel	logical, make parallel?
progress	name of the progress bar to use, see create_progress_bar

Details

Read a tree(s) from a file using the .trmn format. It is faster to read and write tree files using treeman with the .trmn file format. In addition it is possible to encode more information than possible with the Newick, e.g. any taxonomic information and additional slot names added to the tree are recorded in the file.

See Also

writeTrmn, readTree,writeTree, randTree, saveTreeMan, loadTreeMan

Examples

```
library(treeman)
tree <- randTree(10)
writeTrmn(tree, file='test.trmn')
tree <- readTrmn('test.trmn')
file.remove('test.trmn')
```

rmClade *Remove a clade from a tree*

Description

Returns a tree with a clade removed

Usage

```
rmClade(tree, id)
```

Arguments

tree	TreeMan object
id	node ID parent of clade to be removed

Details

Inverse function of `getSubtree()`. Takes a tree and removes a clade based on an internal node specified. Node is specified with `id`, all descending nodes and tips are removed. The resulting tree will replace the missing clade with a tip of `id`.

See Also

addClade, getSubtree, rmTips https://github.com/DomBennett/treeman/wiki/manip-methods

Examples

```
library(treeman)
t1 <- randTree(100)
# remove a clade
t2 <- rmClade(t1, 'n2')
summary(t1)
summary(t2)
```

rmNdmtrx *Remove node matrix*

Description

Return tree with memory heavy node matrix removed.

Usage

```
rmNdmtrx(tree)
```

Arguments

tree	TreeMan object

Details

Potential uses: reduce memory load of a tree, save tree using serialization methods.

See Also

addNdmtrx

Examples

```
# library(treeman)
tree <- randTree(10)
summary(tree)
tree <- rmNdmtrx(tree)
summary(tree)
```

---

rmNodes *Remove nodes from a tree*

Description

Returns a tree with a node ID(s) removed

Usage

```
rmNodes(tree, nids, progress = "none")
```

Arguments

tree	TreeMan object
nids	internal node IDs
progress	name of the progress bar to use, see create_progress_bar

Details

Removes nodes in a tree. Joins the nodes following to the nodes preceding the node to be removed. Creates polytomies. Warning: do not use this function to remove tip nodes, this create a corrupted tree.

See Also

addTip, rmTips, https://github.com/DomBennett/treeman/wiki/manip-methods

Examples

```
library(treeman)
tree <- randTree(10)
tree <- rmNodes(tree, 'n3')
summary(tree)  # tree is now polytmous
```

`rmOtherSlt` *Remove a user-defined slot*

Description

Returns a tree with a user-defined tree slot removed.

Usage

```
rmOtherSlt(tree, slt_nm)
```

Arguments

`tree`	`TreeMan` object
`slt_nm`	name of slot to be removed

Details

A user can specify a new slot using the `setNdSlt()` function or upon reading a tree. This can be removed using this function by specifying the name of the slot to be removed.

See Also

`setNdOther`, `setNdsOther`, https://github.com/DomBennett/treeman/wiki/set-methods

Examples

```
library(treeman)
tree <- randTree(10)
vals <- runif(min=0, max=1, n=tree['nall'])
tree <- setNdsOther(tree, tree['all'], vals, 'confidence')
tree <- updateSlts(tree)
summary(tree)
tree <- rmOtherSlt(tree, 'confidence')
tree <- updateSlts(tree)
summary(tree)
```

rmTips *Remove tips from a tree*

Description

Returns a tree with a tip ID(s) removed

Usage

```
rmTips(tree, tids, drp_intrnl = TRUE, progress = "none")
```

Arguments

tree	TreeMan object
tids	tip IDs
drp_intrnl	Boolean, drop internal branches, default FALSE
progress	name of the progress bar to use, see create_progress_bar

Details

Removes tips in a tree. Set drp_intrnl to FALSE to convert internal nodes into new tips. Warning: do not use this function to remove internal nodes, this create a corrupted tree.

See Also

addTip, rmNodes, https://github.com/DomBennett/treeman/wiki/manip-methods

Examples

```
library(treeman)
tree <- randTree(10)
tree <- rmTips(tree, 't1')
summary(tree)
# running the function using an internal
# node will create a corrupted tree
tree <- rmTips(tree, 'n3')
# run summary() to make sure a change has
# not created a corruption
#summary(tree)
```

saveTreeMan *Save a TreeMan object in serialization format*

Description

TreeMan equivalent to save() but able to handle node matrices.

Usage

```
saveTreeMan(tree, file)
```

Arguments

tree	TreeMan object
file	file path

Details

It is not possible to use save() on TreeMan objects with node matrices. Node matrices are bigmemory matrices and are therefore outside the R environment, see bigmemory documentation for more information. Saving and loading a bigmemory matrix may cause memory issues in R and cause R to crash.

This function can safely store a TreeMan object with and without a node matrix. This function stores the tree using the serialization format and the node matrix as a hidden .csv. Both parts of the tree can be reloaded to an R environment with loadTreeMan(). The hidden node matrix filename is based on the file argument: file + _ndmtrx

Reading and writing trees with saveTreeMan() and loadTreeMan is faster than any of the other read and write functions.

See Also

loadTreeMan, readTree,writeTree, readTrmn, writeTrmn

Examples

```
library(treeman)
tree <- randTree(100, wndmtrx=TRUE)
saveTreeMan(tree, file='test.RData')
rm(tree)
tree <- loadTreeMan(file='test.RData')
file.remove('test.RData', 'testRData_ndmtrx')
```

searchTxnyms *Get node labels based on online taxonomic database*

Description

Return names of each node in tree based on searching tip labels through Global Names Resolver http://resolver.globalnames.org/ in NCBI.

Usage

```
searchTxnyms(tree, cache = FALSE, parent = NULL, clean = TRUE,
  infer = TRUE)
```

Arguments

tree	TreeMan object
cache	T/F, create a local cache of downloaded names?
parent	specify parent of all names to prevent false names
clean	T/F, ensure returned names contain no special characters?
infer	T/F, infer taxonyms for unfound nodes?

Details

For each node, all the descendants are searched, the taxonomic lineages returned and then searched to find the lowest shared name. All the tip labels are searched against a specified taxonomic database through the GNR and NCBI. (So far only tested with NCBI database.) Use the infer argument to ensure a taxonym is returned for all nodes. If infer is true, all nodes without an identifed taxonym adopt the taxonym of their parent.

See Also

taxaResolve, setTxnyms, getNdsFrmTxnyms

Examples

```
tree <- randTree(8)
new_tids <- c("Gallus_gallus", "Aileuropoda_melanoleucha", "Ailurus_fulgens",
"Rattus_rattus", "Mus_musculus", "Gorilla_gorilla", "Pan_trogoldytes", "Homo_sapiens")
tree <- setNdsID(tree, tree['tips'], new_tids)
nd_labels <- searchTxnyms(tree)
print(nd_labels)
```

setAge *Set the age of a tree*

Description

Return a tree with the age altered.

Usage

```
setAge(tree, val)
```

Arguments

tree	TreeMan object
val	new age

Details

Use this function to change the age of a tree. For example, you might want to convert the tree so that its age equals 1. This function will achieve that by modiyfing every branch, while maintaining their relative lengths.

See Also

setPD https://github.com/DomBennett/treeman/wiki/set-methods

Examples

```
library(treeman)
tree <- randTree(10)
tree <- setAge(tree, val=1)
summary(tree)
```

setNdID *Set the ID of a node*

Description

Return a tree with the ID of a node altered.

Usage

```
setNdID(tree, id, val)
```

Arguments

tree	TreeMan object
id	id to be changed
val	new id

Details

IDs cannot be changed directly for the TreeMan class. To change an ID use this function. Warning: all IDs must be unique, avoid spaces in IDs and only use letters, numbers and underscores. Use updateSlts after running.

See Also

setNdsID https://github.com/DomBennett/treeman/wiki/set-methods

Examples

```
library(treeman)
tree <- randTree(10)
tree <- setNdID(tree, 't1', 'heffalump')
tree <- updateSlts(tree)
```

---

setNdOther *Set a user defined slot*

Description

Return a tree with a user defined slot for node ID.

Usage

```
setNdOther(tree, id, val, slt_nm)
```

Arguments

tree	TreeMan object
id	id of the node
val	data for slot
slt_nm	slot name

Details

A user can specify new slots in a tree. Add a new slot with this function by providing a node ID, a value for the new slot and a unique new slot name. Slot names must not be default TreeMan names. The new value can be any data type.

See Also

setNdsOther https://github.com/DomBennett/treeman/wiki/set-methods

Examples

```
library(treeman)
tree <- randTree(10)
tree <- setNdOther(tree, 't1', 1, 'binary_val')
tree <- updateSlts(tree)
(getNdSlt(tree, id='t1', slt_nm='binary_val'))
```

---

setNdsID *Set the IDs of multiple nodes*

Description

Return a tree with the IDs of nodes altered.

Usage

```
setNdsID(tree, ids, vals, parallel = FALSE, progress = "none")
```

Arguments

tree	TreeMan object
ids	ids to be changed
vals	new ids
parallel	logical, make parallel?
progress	name of the progress bar to use, see create_progress_bar

Details

Runs setNdID() over multiple nodes. Warning: all IDs must be unique, avoid spaces in IDs, only use numbers, letters and underscores. Parellizable.

See Also

setNdID https://github.com/DomBennett/treeman/wiki/set-methods

Examples

```
library(treeman)
tree <- randTree(10)
new_ids <- paste0('heffalump_', 1:tree['ntips'])
tree <- setNdsID(tree, tree['tips'], new_ids)
summary(tree)
```

setNdsOther *Set a user defined slot for multiple nodes*

Description

Return a tree with a user defined slot for node IDs.

Usage

```
setNdsOther(tree, ids, vals, slt_nm, parallel = FALSE, progress = "none")
```

Arguments

tree	TreeMan object
ids	id sof the nodes
vals	data for slot
slt_nm	slot name
parallel	logical, make parallel?
progress	name of the progress bar to use, see create_progress_bar

Details

Runs setNdOther() over multiple nodes. Parellizable.

See Also

setNdOther https://github.com/DomBennett/treeman/wiki/set-methods

Examples

```
library(treeman)
tree <- randTree(10)
# e.g. confidences for nodes
vals <- runif(min=0, max=1, n=tree['nall'])
tree <- setNdsOther(tree, tree['all'], vals, 'confidence')
tree <- updateSlts(tree)
summary(tree)
(getNdsSlt(tree, ids=tree['all'], slt_nm='confidence'))
```

setNdSpn *Set the branch length of a specific node*

Description

Return a tree with the span of a node altered.

Usage

```
setNdSpn(tree, id, val)
```

Arguments

tree	TreeMan object
id	id of node whose preceding edge is to be changed
val	new span
...	plyr arguments

Details

Takes a tree, a node ID and a new value for the node's preceding branch length (span).

See Also

setNdsSpn https://github.com/DomBennett/treeman/wiki/set-methods

Examples

```
library(treeman)
tree <- randTree(10)
tree <- setNdSpn(tree, id='t1', val=100)
tree <- updateSlts(tree)
summary(tree)
```

setNdsSpn *Set the branch lengths of specific nodes*

Description

Return a tree with the spans of nodes altered.

Usage

```
setNdsSpn(tree, ids, vals, parallel = FALSE, progress = "none")
```

Arguments

tree	TreeMan object
ids	ids of nodes whose preceding edges are to be changed
vals	new spans
parallel	logical, make parallel?
progress	name of the progress bar to use, see create_progress_bar

Details

Runs setNdSpn over multiple nodes. Parallelizable.

See Also

setNdSpn https://github.com/DomBennett/treeman/wiki/set-methods

Examples

```
library(treeman)
tree <- randTree(10)
# make tree taxonomic
tree <- setNdsSpn(tree, ids=tree['all'], vals=1)
summary(tree)
# remove spns by setting all to 0
tree <- setNdsSpn(tree, ids=tree['all'], vals=0)
summary(tree)
```

---

setPD *Set the phylogenetic diversity*

Description

Return a tree with the phylogenetic diversity altered.

Usage

```
setPD(tree, val)
```

Arguments

tree	TreeMan object
val	new phylogenetic diversity

Details

Use this function to convert the phylogenetic diversity of a tree. For example, you might want to convert the tree so the sum of all branches is 1. This function will achieve that by modiyfing every branch, while maintaining their relative lengths.

See Also

setAge https://github.com/DomBennett/treeman/wiki/set-methods

Examples

```
library(treeman)
tree <- randTree(10)
tree <- setPD(tree, val=1)
summary(tree)
```

---

setTxnyms *Set the txnym slots in a tree*

Description

Return a tree with txnyms added to specified nodes

Usage

```
setTxnyms(tree, txnyms)
```

Arguments

tree	TreeMan object
txnyms	named vector or list

Details

Returns a tree. Specify the taxonomic groups for nodes in a tree by providing a vector or list named by node IDs. Takes output from searchTxnyms. Only letters, numbers and underscores allowed. To remove special characters use regular expressions, e.g. gsub(['a-zA-Z0-9_'], '', txnym)

See Also

taxaResolve, searchTxnyms, getNdsLng, getNdLng, https://github.com/DomBennett/treeman/wiki/set-methods

Examples

```
library(treeman)
data(mammals)
# let's change the txnym for humans
# what's its summary before we change anything?
summary(mammals[['Homo_sapiens']])
# now let's add Hominini
new_txnym <- list('Homo_sapiens'=c('Hominini', 'Homo'))
mammals <- setTxnyms(mammals, new_txnym)
summary(mammals[['Homo_sapiens']])
```

taxaResolve *Resolve taxonmic names online*

Description

Resolve taxonomic names via the Global Names Resolver.

Usage

```
taxaResolve(nms, batch = 100, datasource = 4, genus = TRUE,
  cache = FALSE, parent = NULL)
```

Arguments

nms	vector of names
batch	size of the batches to be queried
datasource	ID number of the datasource
genus	boolean, if true will search against GNR with just the genus name for names that failed to resolve using the full species name
cache	T/F, create a local cache of downloaded names?
parent	specify parent of all names to prevent false names

Details

Returns dataframe containing GNR metadata for each name wames that cannot be resolved are returned as NA. Various datasources are available, see http://resolver.globalnames.org/data_sources for a list and IDs. Default is 4 for NCBI.

See Also

searchTxnyms, setTxnyms, getNdsFrmTxnyms

Examples

```
my_lovely_names <- c ('Gallus gallus', 'Pongo pingu', 'Homo sapiens',
                      'Arabidopsis thaliana', 'Macaca thibetana', 'Bacillus subtilis')
res <- taxaResolve (nms=my_lovely_names)
length(colnames(res))  # 10 different metadata for returned names including original search name
# let's look at the lineages
lineages <- strsplit(as.vector(res$lineage), '\\|')
print(lineages[[6]])  # the bacteria has far fewer taxonomic levels
```

TreeMan-class *TreeMan-class*

Description

S4 class for representing phylogenetic trees as a list of nodes.

Usage

```
## S4 method for signature 'TreeMan,character'
x[[i]]

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

## S4 method for signature 'TreeMan'
as.character(x)

## S4 method for signature 'TreeMan'
show(object)

## S4 method for signature 'TreeMan'
print(x)

## S4 method for signature 'TreeMan'
str(object, max.level = 2L, ...)

## S4 method for signature 'TreeMan'
summary(object)

## S4 method for signature 'TreeMan'
cTrees(x, ...)
```

Arguments

x	TreeMan object
i	node ID or slot name
j	missing
...	additional tree objects
drop	missing
object	TreeMan object
max.level	str() maximum number of levels to show

Details

A TreeMan object holds a list of nodes. The idea of the TreeMan class is to make adding and removing nodes as similar as possible to adding and removing elements in a list. Note that internal nodes and tips are both considered nodes. Trees can be polytomous but not unrooted.

Each node within the TreeMan ndlst contains the following data slots:

- id: character string for the node ID
- txnym: name of taxonomic clade (optional)
- spn: length of the preceding branch
- prid: ID of the immediately preceding node, NULL if root
- ptid: IDs of the immediately connecting nodes

See below in 'Examples' for these methods in use.

Slots

ndlst list of nodes

nds vector of node ids that are internal nodes

nnds numeric of number of internal nodes in tree

tips vector of node ids that are tips

ntips numeric of number of internal nodes in tree

all vector of all node ids

nall numeric of number of all nodes in tree

pd numeric of total branch length of tree

tinds indexes of all tip nodes in tree

prinds indexes of all pre-nodes in tree

wspn logical, do nodes have spans

wtxnyms logical, do nodes have txnyms

ply logical, is tree bifurcating

root character of node id of root, if no root then empty character

updtd logical, if tree slots have been updated since initiation or change

othr_slt_nms vector, character list of additional data slots added to nodes

ndmtrx matrix, T/Fs representing tree structure

See Also

randTree, Node-class, phylo-to-TreeMan, TreeMan-to-phylo

Examples

```
library(treeman)
# Generate random tree
tree <- randTree(10)
# Print to get basic stats
summary(tree)
# Slots....
tree['tips']   # return all tips IDs
tree['nds']    # return all internal node IDs
tree['ntips']  # count all tips
tree['nnds']   # count all internal nodes
tree['root']   # identify root node
tree[['t1']]   # return t1 node object
tree['pd']     # return phylogenetic diversity
tree['ply']    # is polytomous?
# Additional special slots (calculated upon call)
tree['age']   # get tree's age
tree['ultr']  # determine if tree is ultrametric
tree['spns']  # get all the spans of the tree IDs
tree['prids'] # get all the IDs of preceding nodes
tree['ptids'] # get all the IDs of following nodes
tree['txnyms'] # get all the taxonyms of all nodes
# In addition [] can be used for any user-defined slot
# Because all nodes are lists with metadata we can readily
#  get specific information on nodes of interest
nd <- tree[['n2']]
summary(nd)
# And then use the same syntax for the tree
nd['nkids']  # .... nkids, pd, etc.

# Convert to phylo and plot
library(ape)
tree <- as(tree, 'phylo')
plot(tree)
```

TreeMan-to-phylo *Convert TreeMan to phylo*

Description

Return ape's phylo from a TreeMan

See Also

phylo-to-TreeMan, TreeMen-to-multiPhylo multiPhylo-to-TreeMen TreeMan-class

Examples

```
library(treeman)
library(ape)
tree <- randTree(10)
tree <- as(tree, 'phylo')
```

TreeMen-class *TreeMen-class*

Description

S4 class for multiple phylogenetic trees

Usage

```
## S4 method for signature 'TreeMen'
cTrees(x, ...)

## S4 method for signature 'TreeMen,ANY'
x[[i]]

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

## S4 method for signature 'TreeMen'
as.character(x)

## S4 method for signature 'TreeMen'
show(object)

## S4 method for signature 'TreeMen'
str(object, max.level = 2L, ...)

## S4 method for signature 'TreeMen'
print(x)

## S4 method for signature 'TreeMen'
summary(object)
```

Arguments

x	TreeMen object
...	additional tree objects
i	tree index (integer or character)
j	missing
drop	missing

object	TreeMen object
max.level	str() maximum level

Slots

treelst list of TreeMan objects

ntips sum of tips per tree

ntrees total number of trees

See Also

cTrees

TreeMen-to-multiPhylo *Convert TreeMen to multiPhylo*

Description

Return ape's multiPhylo from a TreeMen

See Also

TreeMan-to-phylo, phylo-to-TreeMan, multiPhylo-to-TreeMen TreeMan-class

Examples

```
library(treeman)
library(ape)
trees <- cTrees(randTree(10), randTree(10), randTree(10))
trees <- as(trees, 'multiPhylo')
```

---

twoer *Generate a tree of two tips*

Description

Returns a TreeMan tree with two tips and a root.

Usage

```
twoer(tids = c("t1", "t2"), spns = c(1, 1), rid = "root", root_spn = 0)
```

Arguments

tids	tip IDs
spns	tip spans
rid	root ID
root_spn	root span

Details

Useful for building larger trees with addClade(). Note, a node matrix cannot be added to a tree of two tips.

See Also

TreeMan-class, randTree

Examples

```
library(treeman)
tree <- twoer()
```

---

ultrTree *Make tree ultrametric*

Description

Returns a tree with all tips ending at time 0

Usage

```
ultrTree(tree)
```

Arguments

tree	TreeMan object

Details

Re-calculates the branch lengths in the tree so that all tips are brought to the same time point: all species are extant.

See Also

https://github.com/DomBennett/treeman/wiki/manip-methods

Examples

```
library(treeman)
tree <- randTree(10)
(getDcsd(tree))  # list all extinct tips
tree <- ultrTree(tree)
(getDcsd(tree))  # list all extinct tips
```

---

unblncdTree *Generate an unbalanced tree*

Description

Returns an unbalanced TreeMan tree with n tips.

Usage

```
unblncdTree(n, wndmtrx = FALSE, parallel = FALSE)
```

Arguments

n	number of tips, integer, must be 3 or greater
wndmtrx	T/F add node matrix? Default FALSE.
parallel	T/F run in parallel? Default FALSE.

Details

Equivalent to ape's `stree(type='left')` but returns a TreeMan tree. Tree is always rooted and bifurcating.

See Also

TreeMan-class, randTree, blncdTree

Examples

```
library(treeman)
tree <- unblncdTree(5)
```

updateSlts *Update tree slots after manipulation*

Description

Return tree with updated slots.

Usage

```
updateSlts(tree)
```

Arguments

tree	TreeMan object

Details

Tree slots in the TreeMan object are usually automatically updated. For certain single node manipulations they are not. Run this function to update the slots.

See Also

addNdmtrx, getAge

writeTree *Write a Newick tree*

Description

Creates a Newick tree from a TreeMan object.

Usage

```
writeTree(tree, file, append = FALSE, ndLabels = function(nd) {
  return(NULL) }, parallel = FALSE, progress = "none")
```

Arguments

tree	TreeMan object
file	file path
append	T/F append tree to already existing file
ndLabels	node label function
parallel	logical, make parallel?
progress	name of the progress bar to use, see create_progress_bar

Details

The ndLabels argument can be used to add a user defined node label in the Newick tree. It should take only 1 argument, nd, the node represented as a list. It should only return a single character value that can be added to a newick string.

See Also

https://en.wikipedia.org/wiki/Newick_format, readTree, randTree, readTrmn, writeTrmn, saveTreeMan, loadTreeMan

Examples

```
library(treeman)
tree <- randTree(10)
# write out the tree with node labels as IDs
ndLabels <- function(n) {
n[['id']]
}
writeTree(tree, file='example.tre', ndLabels=ndLabels)
file.remove('example.tre')
```

---

writeTrmn *Write a .trmn tree*

Description

Write to disk a TreeMan or TreeMan object using the .trmn treefile

Usage

```
writeTrmn(tree, file)
```

Arguments

tree	TreeMan object or TreeMen object
file	file path

Details

Write a tree(s) to file using the .trmn format. It is faster to read and write tree files using treeman with the .trmn file format. In addition it is possible to encode more information than possible with the Newick, e.g. any taxonomic information and additional slot names added to the tree are recorded in the file.

See Also

readTrmn, readTree,writeTree, randTree, saveTreeMan, loadTreeMan

Examples

```
library(treeman)
tree <- randTree(10)
writeTrmn(tree, file='test.trmn')
tree <- readTrmn('test.trmn')
file.remove('test.trmn')
```

Index