Package 'TreeTools'

December 19, 2019

Title Create, Modify and Analyse Phylogenetic Trees

Version 0.1.3

License GPL (>= 3)

Copyright Incorporates C/C++ code from: ape by Emmanuel Paradis; phangorn by Klaus Scheip <doi:10.1093/bioinformatics/btq706>

Description Efficient implementations of functions for the creation, modification and analysis of phylogenetic trees.
Applications include: generation of trees with specified shapes;
rooting of trees and extraction of subtrees;
calculation and depiction of node support;
calculation of ancestor-descendant relationships;
import and export of trees from Newick, Nexus (Maddison et al. 1997) <doi:10.1093/sysbio/46.4.590>,
and TNT <http://www.lillo.org.ar/phylogeny/tnt/> formats;
and
analysis of partitions and partition information.

URL https://ms609.github.io/TreeTools,
https://github.com/ms609/TreeTools

BugReports https://github.com/ms609/TreeTools/issues

SystemRequirements C++11

Depends R (>= 3.4.0), ape (>= 5.0)

Imports colorspace, phangorn (>= 2.2.1), R.cache

Suggests knitr, Rcpp, Rdpack, rmarkdown, shiny, testthat

RdMacros Rdpack

LazyData true

ByteCompile true

Encoding UTF-8

Language en-GB

LinkingTo Rcpp

VignetteBuilder knitr

RoxygenNote 6.1.1

NeedsCompilation yes

Author Martin R. Smith [aut, cre, cph]
(<https://orcid.org/0000-0001-5660-1727>),
Emmanuel Paradis [cph] (<https://orcid.org/0000-0003-3092-2199>),
Klaus Schliep [cph] (<https://orcid.org/0000-0003-2941-0161>)

Maintainer Martin R. Smith <martin.smith@durham.ac.uk>

Repository CRAN

Date/Publication 2019-12-19 18:20:07 UTC

R topics documented:

`AddTip` *Add a tip to a phylogenetic tree*

Description

`AddTip` adds a tip to a phylogenetic tree at a specified location.

Usage

```
AddTip(tree, where = sample.int(tree$Nnode * 2 + 2L, size = 1) - 1L,
  label = "New tip")

AddTipEverywhere(tree, label = "New tip", includeRoot = FALSE)
```

Arguments

`tree`	A tree of class `phylo`.
`where`	The node or tip that should form the sister taxon to the new node. To add a new tip at the root, use `where = 0`. By default, the new tip is added to a random edge.
`label`	Character string providing the label to apply to the new tip.
`includeRoot`	Logical; if `TRUE`, the three positions adjacent to the root edge are considered to represent distinct edges.

Details

AddTip extends bind.tree, which cannot handle single-taxon trees.

Value

AddTip returns a tree of class phylo with an additional tip at the desired location.

AddTipEverywhere returns a list of class multiPhylo containing the trees produced by adding label to each edge of tree in turn.

Functions

- AddTipEverywhere: Add a tip to each edge in turn.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

bind.tree

nodelabels

Other tree manipulation: Renumber, SingleTaxonTree, Subtree

Examples

```
plot(tree <- BalancedTree(10))
ape::nodelabels()
ape::nodelabels(15, 15, bg='green')

plot(AddTip(tree, 15, 'NEW_TIP'))

oldPar <- par(mfrow=c(2, 4), mar=rep(0.3, 4), cex=0.9)

backbone <- BalancedTree(4)
additions <- AddTipEverywhere(backbone, includeRoot = TRUE)
xx <- lapply(additions, plot)

par(mfrow=c(2, 3))
additions <- AddTipEverywhere(backbone, includeRoot = FALSE)
xx <- lapply(additions, plot)

par(oldPar)
```

`AllAncestors` *List all ancestral nodes*

Description

`AllAncestors` lists ancestors of each parent node in a tree.

Usage

```
AllAncestors(parent, child)
```

Arguments

`parent`	the first column of the edge matrix of a tree of class `phylo`, i.e. `tree$edge[,1]`.
`child`	the second column of the edge matrix of a tree of class `phylo`, i.e. `tree$edge[,2]`.

Details

Note that the tree's edges must be listed in an order whereby each entry in `tr$edge[,1]` (with the exception of the root) has appeared already in `tr$edge[,2]`.

Value

`AllAncestors` returns a list. Entry *i* contains a vector containing, in order, the nodes encountered when traversing the tree from node *i* to the root node. The last entry of each member of the list is therefore the root node, with the exception of the entry for the root node itself, which is `NULL`.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

Other tree navigation: `AncestorEdge`, `DescendantEdges`, `EdgeAncestry`, `EdgeDistances`, `MRCA`, `NonDuplicateRoot`

Examples

```
tr <- PectinateTree(4)
plot(tr)
ape::tiplabels()
ape::nodelabels()
edge <- tr$edge
AllAncestors(edge[, 1], edge[, 2])
```

ApeTime *Ape Time*

Description

Reads the time that an ape tree was modified from the comment in the Nexus file.

Usage

```
ApeTime(filename, format = "double")
```

Arguments

filename	Character string specifying path to the file
format	Format in which to return the time: 'double' as a sortable numeric; any other value to return a string in the format YYYY-MM-DD hh:mm:ss

Value

ApeTime returns the time that the specified file was created by ape, in the format specified by format.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

as.Newick *Write a phylogenetic tree in Newick format*

Description

Creates a character string describing a phylogenetic tree in Newick format, using R's internal tip numbering. Use RenumberTips to ensure that the internal numbering follows the order you expect.

Usage

```
as.Newick(x)

## S3 method for class 'phylo'
as.Newick(x)

## S3 method for class 'list'
as.Newick(x)

## S3 method for class 'multiPhylo'
as.Newick(x)
```

Arguments

x	Object to convert to Newick format. See Usage section for supported classes.

Value

`as.Newick` returns a character string representing `tree` in Newick format.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

- NewickTree
- RenumberTips
- ape::write.tree

Examples

```
trees <- list(BalancedTree(1:8), PectinateTree(8:1))
trees <- lapply(trees, RenumberTips, 1:8)
as.Newick(trees)
```

---

as.Splits *As splits*

Description

Converts a phylogenetic tree to an array of bipartition splits.

Usage

```
as.Splits(x, tipLabels = NULL, ...)

## S3 method for class 'phylo'
as.Splits(x, tipLabels = NULL, asSplits = TRUE, ...)

## S3 method for class 'multiPhylo'
as.Splits(x, tipLabels = x[[1]]$tip.label,
  asSplits = TRUE, ...)

## S3 method for class 'Splits'
as.Splits(x, tipLabels = NULL, ...)

## S3 method for class 'list'
as.Splits(x, tipLabels = NULL, asSplits = TRUE, ...)
```

```
## S3 method for class 'logical'
as.Splits(x, tipLabels = NULL, ...)

## S3 method for class 'Splits'
as.logical(x, tipLabels = NULL, ...)
```

Arguments

x	Object to convert into splits: perhaps a tree of class phylo. If a logical matrix is provided, each row will be considered as a separate split.
tipLabels	Character vector specifying sequence in which to order tip labels. Label order must (currently) match to combine or compare separate Splits objects.
...	Presently unused.
asSplits	Logical specifying whether to return a Splits object, or an unannotated two-dimensional array (useful where performance is paramount).

Value

as.Splits returns an object of class Splits, or (if asSplits = FALSE) a two-dimensional array of 32-bit integers, which each bit specifying whether a tip is a member of the split. Splits are named according to the node that defines them.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

Other Splits operations: NSplits, NTip, TipLabels, TipsInSplits, in.Splits, match.Splits

Examples

```
splits <- as.Splits(BalancedTree(letters[1:6]))
summary(splits)
TipsInSplits(splits)
summary(!splits)
TipsInSplits(!splits)

length(splits + !splits)
length(unique(splits + !splits))

summary(c(splits[[2:3]], !splits[[1:2]]))

moreSplits <- as.Splits(PectinateTree(letters[6:1]), tipLabel = splits)
print(moreSplits, details = TRUE)
match.Splits(splits, moreSplits)
in.Splits(moreSplits, splits)
```

BalancedTree *Generate a Balanced Tree*

Description

Generates a balanced (symmetrical) binary tree with the specified tip labels.

Usage

```
BalancedTree(tips)
```

Arguments

tips	An integer specifying the number of tips, or a character vector naming the tips, or any other object from which tip labels can be extracted with function TipLabels.

Value

A tree of class phylo.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

Other tree generation functions: NJTree, PectinateTree, RandomTree, SingleTaxonTree, TreeNumber

Examples

```
plot(BalancedTree(LETTERS[1:10]))
```

brewer *Brewer palettes*

Description

A list of eleven Brewer palettes containing one to eleven colours that are readily distinguished by colourblind viewers, followed by a twelfth 12-colour palette adapted for colour blindness.

Usage

```
brewer
```

Format

An object of class `list` of length 12.

Source

- ColourBrewer2.org
- Martin Krzywinski

Examples

```
data("brewer", package="TreeTools")
plot(0, type='n', xlim=c(1, 12), ylim=c(12, 1),
     xlab = 'Colour', ylab='Palette')
for (i in seq_along(brewer)) text(seq_len(i), i, col=brewer[[i]])
```

`CharacterInformation` *Character information content*

Description

Calculates the phylogenetic information content of a given character.

Usage

```
CharacterInformation(tokens)
```

Arguments

`tokens`	Character vector specifying the tokens assigned to each taxon for a character. Example: `c(0,0,0,1,1,1,'?','-')`. Note that ambiguous tokens such as `(01)` are not supported, and should be replaced with `?`.‘

Value

`CharacterInformation` returns a numberic specifying the phylogenetic information content of the character, in bits.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

Other split information functions: `SplitInformation`, `TreesMatchingSplit`, `UnrootedTreesMatchingSplit`

`CollapseNode` *Collapse nodes on a phylogenetic tree*

Description

Collapses specified nodes or edges on a phylogenetic tree, resulting in polytomies.

Usage

```
CollapseNode(tree, nodes)

CollapseEdge(tree, edges)
```

Arguments

`tree`	A tree of class `phylo`.
`nodes, edges`	Integer vector specifying the nodes or edges in the tree to be dropped. (Use `nodelabels` or `edgelabels` to view numbers on a plotted tree.)

Value

`CollapseNode` and `CollapseEdge` return a tree of class `phylo`, corresponding to `tree` with the specified nodes or edges collapsed. The length of each dropped edge will (naively) be added to each descendant edge.

Author(s)

Martin R. Smith

Examples

```
library(ape)
set.seed(1)
oldPar <- par(mfrow=c(2, 1), mar=rep(0.5, 4))

tree <- rtree(7)
plot(tree)
nodelabels()
edgelabels(round(tree$edge.length, 2), cex=0.6, frame='n', adj=c(1, -1))

newTree <- CollapseNode(tree, c(12, 13))
plot(newTree)
nodelabels()
edgelabels(round(newTree$edge.length, 2), cex=0.6, frame='n', adj=c(1, -1))

par(oldPar)
```

`ConsensusWithout` *Consensus without taxa*

Description

Displays a consensus plot with selected taxa excluded.

Usage

```
ConsensusWithout(trees, tip, ...)

MarkMissing(tip, position = "bottomleft", ...)
```

Arguments

`trees`	A list of phylogenetic trees, of class `multiPhylo` or `list`.
`tip`	A character vector specifying the names (or numbers) of tips to drop (using ape::drop.tip).
`...`	Additional parameters to pass on to ape::consensus or legend.
`position`	Where to plot the missing taxa. See legend for options.

Details

A useful way to gain resolution if a few wildcard taxa obscure a consistent set of relationship.

Value

`ConsensusWithout` returns a consensus tree (of class `phylo`) without the excluded taxa.

Functions

- `MarkMissing`: Adds labels for taxa omitted from a plotted consensus tree.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

Martin R. Smith (martin.smith@durham.ac.uk)

Examples

```
oldPar <- par(mfrow=c(1, 2), mar=rep(0.5, 4))

# Two trees differing only in placement of tip 2:
trees <- as.phylo(c(0, 53), 6)
plot(trees[[1]])
plot(trees[[2]])

# Strict consensus lacks resolution:
```

```
plot(ape::consensus(trees))

# But omitting tip two reveals shared structure in common:
plot(ConsensusWithout(trees, 't2'))
MarkMissing('t2')

par(oldPar)
```

DescendantEdges *Descendant Edges*

Description

Quickly identifies edges that are 'descended' from a particular edge in a tree

Usage

```
DescendantEdges(edge, parent, child, nEdge = length(parent))

AllDescendantEdges(parent, child, nEdge = length(parent))
```

Arguments

edge	number of the edge whose child edges are required.
parent	the first column of the edge matrix of a tree of class phylo, i.e. tree$edge[,1].
child	the second column of the edge matrix of a tree of class phylo, i.e. tree$edge[,2].
nEdge	number of edges (calculated from length(parent) if not supplied).

Value

DescendantEdges returns a logical vector stating whether each edge in turn is a descendant of the specified edge (or the edge itself).

AllDescendantEdges returns a matrix of class logical, with row N specifying whether each edge is a descendant of edge N (or the edge itself).

Functions

- AllDescendantEdges: Quickly identifies edges that are 'descended' from each edge in a tree.

See Also

Other tree navigation: AllAncestors, AncestorEdge, EdgeAncestry, EdgeDistances, MRCA, NonDuplicateRoot

DoubleFactorial *Double Factorial*

Description

Double Factorial

Usage

```
DoubleFactorial(n)

LogDoubleFactorial(n)

LogDoubleFactorial.int(n)
```

Arguments

n	Vector of integers.

Value

Returns the double factorial, n x (n - 2) x (n - 4) x (n - 6) x ...

Functions

- `LogDoubleFactorial`: Returns the logarithm of the double factorial.
- `LogDoubleFactorial.int`: Slightly faster, when x is known to be length one and below 50001

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

Other Double factorial: `doubleFactorials`, `logDoubleFactorials`

Examples

```
{
DoubleFactorial (-4:0) # Return 1 if n < 2
DoubleFactorial (2) # 2
DoubleFactorial (5) # 1 x 3 x 5
exp(LogDoubleFactorial.int (8)) # 2 x 4 x 6 x 8

}
```

doubleFactorials *Double factorials*

Description

A vector with pre-calculated values of double factorials up to 300!!, and the logarithms of double factorials up to 50 000!!.

Usage

```
doubleFactorials
```

Format

An object of class `numeric` of length 300.

Details

301!! is too large to store as an integer; use `logDoubleFactorials` instead.

See Also

Other Double factorial: `DoubleFactorial`, `logDoubleFactorials`

EdgeAncestry *Edge ancestry*

Description

Quickly identify edges that are 'ancestral' to a particular edge in a tree.

Usage

```
EdgeAncestry(edge, parent, child, stopAt = (parent == min(parent)))
```

Arguments

`edge`	Integer specifying the number of the edge whose child edges should be returned.
`parent`	the first column of the edge matrix of a tree of class `phylo`, i.e. `tree$edge[,1]`.
`child`	the second column of the edge matrix of a tree of class `phylo`, i.e. `tree$edge[,2]`.
`stopAt`	number of the edge at which the search should terminate; defaults to the root edges.

Value

EdgeAncestry returns a logical vector stating whether each edge in turn is a descendant of the specified edge.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

Other tree navigation: AllAncestors, AncestorEdge, DescendantEdges, EdgeDistances, MRCA, NonDuplicateRoot

Examples

```
tree <- PectinateTree(6)
plot(tree)
ape::edgelabels()
parent <- tree$edge[, 1]
child <- tree$edge[, 2]
EdgeAncestry(7, parent, child)
which(EdgeAncestry(7, parent, child, stopAt = 4))
```

EdgeDistances *Distance between edges*

Description

Number of nodes that must be traversed to navigate from each edge to each other edge within a tree

Usage

```
EdgeDistances(tree)
```

Arguments

tree	A tree of class phylo.

Value

A symmetrical matrix listing the number of edges that must be traversed to travel from each numbered edge to each other. The two edges straddling the root of a rooted tree are counted as a single edge. Add a 'root' tip using AddTip if the position of the root is significant.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

Other tree navigation: AllAncestors, AncestorEdge, DescendantEdges, EdgeAncestry, MRCA, NonDuplicateRoot

Examples

```
tree <- BalancedTree(5)
plot(tree)
ape::edgelabels()

EdgeDistances(tree)
```

EnforceOutgroup *Force taxa to form an outgroup*

Description

Given a tree or a list of taxa, rearrange the ingroup and outgroup taxa such that the two are sister taxa across the root, without changing the relationships within the ingroup or within the outgroup.

Usage

```
EnforceOutgroup(tree, outgroup)
```

Arguments

tree	Either: a tree of class phylo; or a character vector listing the names of all the taxa in the tree, from which a random tree will be generated.
outgroup	Character vector containing the names of taxa to include in the outgroup.

Value

EnforceOutgroup returns a tree of class phylo where all outgroup taxa are sister to all remaining taxa, without modifying the ingroup topology.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

Examples

```
tree <- EnforceOutgroup(letters[1:9], letters[1:3])
plot(tree)
```

`in.Splits` *Splits in Splits object*

Description

`in.Splits` is an equivalent to `%in%` that can be applied to objects of class `Splits`.

Usage

```
in.Splits(x, table, incomparables = NULL)
```

Arguments

`x, table`	Object of class `Splits`.
`incomparables`	A vector of values that cannot be matched. Any value in `x` matching a value in this vector is assigned the `nomatch` value. For historical reasons, `FALSE` is equivalent to `NULL`.

Value

A logical vector specifying which of the splits in `x` are present in `table`.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

Other Splits operations: `NSplits`, `NTip`, `TipLabels`, `TipsInSplits`, `as.Splits`, `match.Splits`

Examples

```
splits1 <- as.Splits(BalancedTree(7))
splits2 <- as.Splits(PectinateTree(7))

in.Splits(splits1, splits2)
```

`Lobo.data` *Raw data from Zhang et al. 2016*

Description

Raw data from Zhang et al. 2016

Modified so that absences are treated appropriately:

Character 7 inapplicable to absent where cephalic shield (char 3) is absent

Character 40 inapplicable to absent where paired appendages absent

Character 46 inapplicable to absent

Character 64 76: stet; trunk annulations / limbs may primitively have been papillate or non-papillate

Character 69 a good case that the fusion of flaps with endopods is secondary; thus inapplicable to absent where 67 is applicable

Character 72 inapplicable to absent where appendages are present

Character 77 inapplicable to absent where papillae applicable, as spine is a secondary elaboration of papillae

Character 78 inapplicable to absent

Character 79 stet, as possible that limbs evolved by extension of plate-like exoskeletal element

Character 80 inapplicable to absent, on assumption that ancestral claws were simple

Characters 83, 84, 86, 87, 92 inapplicable to absent; an obvious elaboration

Character 93 inapplicable to absent; ancestrally undifferentiated by definition?

Character 96 inapplicable to absent; ancestrally undifferentiated by definition. Ambiguous in taxa that lack claws as rotation would not be observed.]

Usage

```
Lobo.data
```

Format

An object of class `list` of length 48.

Source

Zhang X, Smith MR, Yang J, Hou J (2016). "Onychophoran-like musculature in a phosphatized Cambrian lobopodian." *Biology Letters*, **12**(9), 20160492. doi: 10.1098/rsbl.2016.0492, http://rsbl.royalsocietypublishing.org/content/12/9/20160492.

Lobo.phy *Data from Zhang et al. 2016 in phyDat format*

Description

Data from Zhang et al. 2016 in phyDat format

Modified so that absences are treated appropriately:

Character 7 inapplicable to absent where cephalic shield (char 3) is absent

Character 40 inapplicable to absent where paired appendages absent

Character 46 inapplicable to absent

Character 64 76: stet; trunk annulations / limbs may primitively have been papillate or non-papillate

Character 69 a good case that the fusion of flaps with endopods is secondary; thus inapplicable to absent where 67 is applicable

Character 72 inapplicable to absent where appendages are present

Character 77 inapplicable to absent where papillae applicable, as spine is a secondary elaboration of papillae

Character 78 inapplicable to absent

Character 79 stet, as possible that limbs evolved by extension of plate-like exoskeletal element

Character 80 inapplicable to absent, on assumption that ancestral claws were simple

Characters 83, 84, 86, 87, 92 inapplicable to absent; an obvious elaboration

Character 93 inapplicable to absent; ancestrally undifferentiated by definition?

Character 96 inapplicable to absent; ancestrally undifferentiated by definition. Ambiguous in taxa that lack claws as rotation would not be observed.]

Usage

```
Lobo.phy
```

Format

An object of class `phyDat` of length 48.

Source

Zhang X, Smith MR, Yang J, Hou J (2016). "Onychophoran-like musculature in a phosphatized Cambrian lobopodian." *Biology Letters*, **12**(9), 20160492. doi: 10.1098/rsbl.2016.0492, http://rsbl.royalsocietypublishing.org/content/12/9/20160492.

logDoubleFactorials *Natural logarithms of double factorials*

Description

A vector with pre-calculated values of double factorials up to 50 000!!.

Usage

```
logDoubleFactorials
```

Format

An object of class `numeric` of length 50000.

See Also

Other Double factorial: `DoubleFactorial`, `doubleFactorials`

match.Splits *Match splits*

Description

Equivalent of `match` for `Splits` objects.

Usage

```
match.Splits(x, table, nomatch = NA_integer_, incomparables = NULL)
```

Arguments

`x, table`	Object of class `Splits`.
`nomatch`	The value to be returned in the case where no match is found.
`incomparables`	A vector of values that cannot be matched. Any value in `x` matching a value in this vector is assigned the `nomatch` value. For historical reasons, `FALSE` is equivalent to `NULL`.

Value

An integer vector specifying the position in `table` that matches each element in `x`, or `nomatch` if no match is found.

See Also

Other Splits operations: `NSplits`, `NTip`, `TipLabels`, `TipsInSplits`, `as.Splits`, `in.Splits`

Examples

```
splits1 <- as.Splits(BalancedTree(7))
splits2 <- as.Splits(PectinateTree(7))

match.Splits(splits1, splits2)
```

MRCA *Most Recent Common Ancestor*

Description

What is the last common ancestor of the specified tips?

Usage

```
MRCA(tip1, tip2, ancestors)
```

Arguments

`tip1, tip2`	Integer specifying index of tips whose most recent common ancestor should be found.
`ancestors`	Output of `AllAncestors` for the tree in question

Value

`MRCA` returns an integer specifying the node number of the last common ancestor of `tip1` and `tip2`.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

Other tree navigation: `AllAncestors`, `AncestorEdge`, `DescendantEdges`, `EdgeAncestry`, `EdgeDistances`, `NonDuplicateRoot`

N1Spr *Number of trees one SPR step away*

Description

N1Spr calculates the number of trees one subtree prune-and-regraft operation away from a binary input tree using the formula given by Allen and Steel (2001).

Usage

```
N1Spr(n)

IC1Spr(n)
```

Arguments

n	Integer vector specifying the number of tips in a tree.

Details

IC1Spr calculates the information content of trees at this distance: i.e. the entropy corresponding to the proportion of all possible *n*-tip trees whose SPR distance is at most one from a specified tree.

Value

N1SPR returns an integer vector.

IC1SPR returns an numeric vector.

Functions

- IC1Spr: Information content of trees 0 or 1 SPR step from tree with n tips.

References

Allen BL, Steel MA (2001). "Subtree transfer operations and their induced metrics on evolutionary trees." *Annals of Combinatorics*, **5**(1), 1–15. ISSN 0218-0006, doi: 10.1007/s0002600180068.

Examples

```
N1Spr(4:6)
IC1Spr(5)
```

NewickTree *Newick Tree*

Description

Writes a tree in Newick format. This differs from ape's `write.tree` in the encoding of spaces as spaces, rather than underscores.

Usage

```
NewickTree(tree)
```

Arguments

`tree`	A tree of class `phylo`.

Value

`NewickTree` returns a character string denoting `tree` in Newick format.

See Also

`as.Newick`

Examples

```
NewickTree(BalancedTree(6))
```

NJTree *Neighbour Joining Tree*

Description

Generates a rooted neighbour joining tree, with no edge lengths.

Usage

```
NJTree(dataset)
```

Arguments

`dataset`	A phylogenetic data matrix of class `phyDat`, whose names correspond to the labels of any accompanying tree.

Value

NJTree returns an object of class phylo.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

Other tree generation functions: BalancedTree, PectinateTree, RandomTree, SingleTaxonTree, TreeNumber

Examples

```
data('Lobo')
NJTree(Lobo.phy)
```

NPartitionPairs *Distributions of tips consistent with a partition pair*

Description

Number of terminal arrangements matching a specified configuration of two partitions.

Usage

```
NPartitionPairs(configuration)
```

Arguments

configuration	Integer vector of length four specifying the number of terminals that occur in both (1) splits A1 and A2; (2) splits A1 and B2; (3) splits B1 and A2; (4) splits B1 and B2.

Details

Consider partitions that divide eight terminals, labelled A to H.

Bipartition 1:	ABCD:EFGH	A1 = ABCD	B1 = EFGH
Bipartition 2:	ABE:CDFGH	A2 = ABE	B2 = CDFGH

This can be represented by an association matrix:

	A2	*B2*
A1	AB	C
B1	E	FGH

The cells in this matrix contain 2, 1, 1 and 3 terminals respectively; this four-element vector (`c(2,1,1,3)`) is the `configuration` implied by this pair of bipartition splits.

Value

The number of ways to distribute `sum(configuration)` taxa according to the specified pattern.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

Examples

```
NPartitionPairs(c(2, 1, 1, 3))
```

`NRooted` *Number of rooted/unrooted trees*

Description

These functions return the number of rooted or unrooted trees consistent with a given pattern of splits.

Usage

```
NRooted(tips)

NUnrooted(tips)

LnUnrooted(tips)

LnUnrooted.int(tips)

LnRooted(tips)

LnRooted.int(tips)

LnUnrootedSplits(splits)

NUnrootedSplits(splits)

LnUnrootedMult(splits)

NUnrootedMult(splits)
```

Arguments

tips	Integer specifying the number of tips.
splits	Integer vector listing the number of taxa in each tree bipartition.

Details

Functions starting N return the number of rooted or unrooted trees, functions starting Ln provide the natural logarithm of this number. Calculations follow Carter et al. 1990, Theorem 2.

Functions

- NUnrooted: Number of unrooted trees
- LnUnrooted: Log Number of unrooted trees
- LnUnrooted.int: Log Number of unrooted trees (as integer)
- LnRooted: Log Number of rooted trees
- LnRooted.int: Log Number of rooted trees (as integer)
- LnUnrootedSplits: Log number of unrooted trees
- NUnrootedSplits: Number of unrooted trees
- LnUnrootedMult: Log unrooted mult
- NUnrootedMult: Number of unrooted trees (mult)

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

References

Carter M, Hendy M, Penny D, SzÃ©kely LA, Wormald NC (1990). "On the distribution of lengths of evolutionary trees." *SIAM Journal on Discrete Mathematics*, **3**(1), 38–47. doi: 10.1137/0403005, http://epubs.siam.org/doi/abs/10.1137/0403005.

Examples

```
NRooted(10)
NUnrooted(10)
LnRooted(10)
LnUnrooted(10)
# Number of trees consistent with a character whose states are
# 00000 11111 222
NUnrootedMult(c(5,5,3))
```

NSplits *Number of distinct partitions*

Description

How many unique bipartition splits occur in a tree or object?

Usage

```
NSplits(x)

NPartitions(x)

## S3 method for class 'phylo'
NSplits(x)

## S3 method for class 'multiPhylo'
NSplits(x)

## S3 method for class 'list'
NSplits(x)

## S3 method for class 'Splits'
NSplits(x)

## S3 method for class 'numeric'
NSplits(x)
```

Arguments

x	A phylogenetic tree of class phylo, or a list of such trees (of class list or multiPhylo), or a Splits object, or a vector of integers.

Value

NSplits returns an integer specifying the number of partitions in the specified objects, or in a rooted tree with n tips.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

Other Splits operations: NTip, TipLabels, TipsInSplits, as.Splits, in.Splits, match.Splits

Examples

```
NSplits(8L)
NSplits(PectinateTree(8))
NSplits(as.Splits(BalancedTree(8)))
```

NTip *Number of tips in a phylogenetic tree*

Description

Extends ape's function Ntip to handle objects of class Splits and list.

Usage

```
NTip(phy)

## S3 method for class 'Splits'
NTip(phy)

## S3 method for class 'list'
NTip(phy)

## S3 method for class 'phylo'
NTip(phy)

## S3 method for class 'multiPhylo'
NTip(phy)
```

Arguments

phy	Object to count.

Value

NTip returns an integer specifying the number of tips in each object in phy.

See Also

Other Splits operations: NSplits, TipLabels, TipsInSplits, as.Splits, in.Splits, match.Splits

PectinateTree *Generate a Pectinate Tree*

Description

Generates a pectinate (caterpillar) tree with the specified tip labels.

Usage

```
PectinateTree(tips)
```

Arguments

tips	An integer specifying the number of tips, or a character vector naming the tips, or any other object from which tip labels can be extracted with function TipLabels.

Value

PectinateTree and BalancedTree each return a binary tree of class phylo of the specified shape.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

Other tree generation functions: BalancedTree, NJTree, RandomTree, SingleTaxonTree, TreeNumber

Examples

```
plot(PectinateTree(LETTERS[1:10]))
```

PhyToString *phyDat to String*

Description

Extract character data from a phyDat object as a string.

Usage

```
PhyToString(phy, parentheses = "{", collapse = "", ps = "",
  useIndex = TRUE, byTaxon = TRUE, concatenate = TRUE)

PhyDatToString(phy, parentheses = "{", collapse = "", ps = "",
  useIndex = TRUE, byTaxon = TRUE, concatenate = TRUE)

PhydatToString(phy, parentheses = "{", collapse = "", ps = "",
  useIndex = TRUE, byTaxon = TRUE, concatenate = TRUE)
```

Arguments

phy	An object of class phyDat
parentheses	Character specifying format of parentheses with which to surround ambiguous tokens. Choose from: { (default), [, (, <.
collapse	Character specifying text, perhaps ,, with which to separate multiple tokens within parentheses
ps	Character specifying text, perhaps ;, to append to the end of the string
useIndex	Logical (default: TRUE) specifying whether to print duplicate characters multiple times, as they appeared in the original matrix
byTaxon	Logical. If TRUE, write one taxon followed by the next. If FALSE, write one character followed by the next.
concatenate	Logical specifying whether to concatenate all characters/taxa into a single string, or to return a separate string for each entry.

Value

PhyToString returns a character vector listing a text representation of the phylogenetic character state for each taxon in turn.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

- StringToPhyDat

Examples

```
fileName <- paste0(system.file(package='TreeTools'),
                   '/extdata/input/dataset.nex')
phyDat <- ReadAsPhyDat(fileName)
PhyToString(phyDat, concatenate = FALSE)
```

print.TreeNumber *Print TreeNumber object*

Description

S3 method for objects of class TreeNumber.

Usage

```
## S3 method for class 'TreeNumber'
print(x, ...)
```

Arguments

x	Object of class TreeNumber.
...	Additional arguments for consistency with S3 method (unused).

RandomTree *Generate a random tree topology*

Description

Generates a binary tree with a random topology on specified tips, optionally rooting the tree on a given tip.

Usage

```
RandomTree(tips, root = FALSE)
```

Arguments

tips	An integer specifying the number of tips, or a character vector naming the tips, or any other object from which tip labels can be extracted with function TipLabels.
root	Tip to use as root (if desired; FALSE otherwise)

Value

RandomTree returns a random tree of class phylo, with the specified tips, and no branch lengths specified.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

Other tree generation functions: BalancedTree, NJTree, PectinateTree, SingleTaxonTree, TreeNumber

Examples

```
RandomTree(letters[1:5])

data('Lobo')
RandomTree(Lobo.phy)
```

ReadCharacters *Read characters from Nexus file*

Description

Parses a Nexus file, reading character states and names.

Usage

```
ReadCharacters(filepath, character_num = NULL, session = NULL)

ReadTntCharacters(filepath, character_num = NULL, session = NULL)

ReadAsPhyDat(filepath)

ReadTntAsPhyDat(filepath)

PhyDat(dataset)
```

Arguments

filepath	character string specifying location of file
character_num	Index of character(s) to return. NULL, the default, returns all characters.
session	(optionally) a Shiny session with a numericInput named character_num whose maximum should be updated.
dataset	list of taxa and characters, in the format produced by read.nexus.data: a list of sequences each made of a single vector of mode character, and named with the taxon name.

Details

Tested with nexus files downloaded from MorphoBank with the "no notes" option, but should also work more generally.

Please report incorrectly parsed files.

Value

A matrix whose row names correspond to tip labels, and column names correspond to character labels, with the attribute `state.labels` listing the state labels for each character; or a character string explaining why the character cannot be returned.

Functions

- `ReadTntCharacters`: Read characters from TNT file
- `ReadAsPhyDat`: Read Nexus characters as phyDat object.
- `ReadTntAsPhyDat`: Read TNT characters as phyDat object.
- `PhyDat`: A convenient wrapper for **phangorn**'s `phyDat`, which converts a *list* of morphological characters into a `phyDat` object. If your morphological characters are in the form of a *matrix*, perhaps because they have been read using `read.table`, try MatrixToPhyDat instead.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

References

Maddison DR, Swofford DL, Maddison WP (1997). "Nexus: an extensible file format for systematic information." *Systematic Biology*, **46**, 590–621. doi: 10.1093/sysbio/46.4.590.

Examples

```
fileName <- paste0(system.file(package='TreeTools'),
                   '/extdata/input/dataset.nex')
ReadCharacters(fileName)
```

`ReadTntTree` *Parse TNT Tree*

Description

Reads a tree from TNT's parenthetical output.

Usage

```
ReadTntTree(filename, relativePath = NULL, keepEnd = 1L,
  tipLabels = NULL)

TNTText2Tree(treeText)
```

Arguments

filename	character string specifying path to TNT .tre file, relative to the R working directory (visible with getwd()).
relativePath	(discouraged) character string specifying location of the matrix file used to generate the TNT results, relative to the current working directory. Taxon names will be read from this file if they are not specified by tipLabels.
keepEnd	(optional, default 1) integer specifying how many elements of the file path to conserve when creating relative path (see examples).
tipLabels	(optional) character vector specifying the names of the taxa, in the sequence that they appear in the TNT file. If not specified, taxon names will be loaded from the data file linked in the first line of the .tre file specified in filename.
treeText	Character string describing a tree, in the parenthetical format output by TNT.

Details

TNT is software for parsimony analysis. Whilst its implementation of tree search is extremely rapid, analysis of results in TNT is made difficult by its esoteric and scantly documented scripting language.

ReadTNTTree aims to aid the user by facilitating the import of trees generated in TNT into R for further analysis.

The function depends on tree files being saved by TNT in parenthetical notation, using the TNT command tsav*. Trees are easiest to load into R if taxa have been saved using their names (TNT command taxname=). In this case, the TNT .tre file contains tip labels and can be parsed directly. The downside is that the uncompressed .tre files will have a larger file size.

ReadTNTTree can also read .tre files in which taxa have been saved using their numbers (taxname-). Such files contain a hard-coded link to the matrix file that was used to generate the trees, in the first line of the .tre file. This poses problems for portability: if the matrix file is moved, or the .tre file is accessed on another computer, the taxon names may be lost. As such, it is important to check that the matrix file exists in the expected location – if it does not, either use the relativePath argument to point to its new location, or specify tipLabels to manually specify the tip labels.

Value

ReadTNTTree returns a tree of class phylo, corresponding to the tree in filename.

Functions

- TNTText2Tree: Converts text representation of a tree in TNT to an object of class phylo.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

Martin R. Smith (martin.smith@durham.ac.uk)

Examples

```
# In the examples below, TNT has read a matrix from
# "c:/TreeTools/input/dataset.nex"
# The results of an analysis were written to
# "c:/TreeTools/output/results1.tnt"
#
# results1.tnt will contain a hard-coded reference to
# "c:/TreeTools/input/dataset.nex".

# On the original machine (but not elsewhere), it would be possible to read
# this hard-coded reference from results.tnt:
# ReadTntTree('output/results1.tnt')

# These datasets are provided with the `TreeTools` package, which will
# probably not be located at c:/TreeTools on your machine:

oldWD <- getwd() # Remember the current working directory
setwd(system.file(package = 'TreeTools'))

# If taxon names were saved within the file (using `taxname=` in TNT),
# then our job is easy:
ReadTntTree('extdata/output/named.tre')

# But if taxa were compressed to numbers (using `taxname-`), we need to
# look up the original matrix in order to dereference the tip names.
#
# We need to extract the relevant file path from the end of the
# hard-coded path in the original file.
#
# We are interested in the last two elements of
# c:/TreeTools/input/dataset.nex
#                2      1
#
# '.' means "relative to the current directory"
ReadTntTree('extdata/output/numbered.tre', './extdata', 2)

# If working in a lower subdirectory
setwd('./extdata/otherfolder')

# then it will be necessary to navigate up the directory path with '..':
ReadTntTree('../output/numbered.tre', '..', 2)


setwd(oldWD) # Restore original working directory
```

Renumber *Renumber a tree's nodes and tips*

Description

Renumber numbers the nodes and tips in a tree to conform with the phylo standards.

Usage

```
Renumber(tree)
```

Arguments

tree	A tree of class phylo.

Value

This function returns a tree of class phylo.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

Other tree manipulation: AddTip, SingleTaxonTree, Subtree

Examples

```
tree <- RandomTree(letters[1:10])
Renumber(tree)
```

RenumberTips *Renumber a tree's tips*

Description

RenumberTips(tree,tipOrder) sorts the tips of a phylogenetic tree such that the indices in tree$edge[,2] correspond to the order of tips given in tipOrder.

Usage

```
RenumberTips(tree, tipOrder)

## S3 method for class 'phylo'
RenumberTips(tree, tipOrder)

## S3 method for class 'multiPhylo'
RenumberTips(tree, tipOrder)

## S3 method for class 'list'
RenumberTips(tree, tipOrder)
```

Arguments

tree	A tree of class phylo.
tipOrder	A character vector containing the values of tree$tip.label in the desired sort order, or an object (perhaps of class phylo or Splits) with tip labels.

Value

RenumberTips returns tree1, with the tips' internal representation numbered to match tipOrder.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

Examples

```
data('Lobo') # Loads the phyDat object Lobo.phy
tree <- RandomTree(Lobo.phy)
tree <- RenumberTips(tree, names(Lobo.phy))
```

RootTree *Root a phylogenetic tree*

Description

Roots a tree on the smallest clade containing the specified tips.

Usage

```
RootTree(tree, outgroupTips)

RootOnNode(tree, node, resolveRoot = FALSE)
```

Arguments

tree	A tree of class phylo.
outgroupTips	Character vector specifying the names of the tips to include in the outgroup.
node	integer specifying node (internal or tip) to set as the root.
resolveRoot	logical specifying whether to resolve the root node.

Value

RootTree returns a tree of class phylo, rooted on the smallest clade that contains the specified tips.

RootOnNode returns a tree of class phylo, rooted on the requested node and ordered in Preorder.

Functions

- RootOnNode: Roots a tree on a specified internal node.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

- ape::root
- EnforceOutgroup

Examples

```
tree <- PectinateTree(8)
plot(tree)
ape::nodelabels()

plot(RootTree(tree, c('t6', 't7')))

plot(RootOnNode(tree, 12))
plot(RootOnNode(tree, 2))
```

SingleTaxonTree *SingleTaxonTree*

Description

Single taxon tree

Usage

```
SingleTaxonTree(label)
```

Arguments

label	a character vector specifying the label of the tip.

Details

Create a phylogenetic 'tree' that comprises a single taxon.

Value

SingleTaxonTree returns a phylo object containing a single tip with the specified label.

See Also

Other tree manipulation: AddTip, Renumber, Subtree

Other tree generation functions: BalancedTree, NJTree, PectinateTree, RandomTree, TreeNumber

Examples

```
SingleTaxonTree('Homo_sapiens')
plot(SingleTaxonTree('root') + BalancedTree(4))
```

SortTree *Sort tree*

Description

Sorts each node into a consistent order, so similar trees look visually similar.

Usage

```
SortTree(tree)
```

Arguments

tree	A tree of class phylo.

Value

SortTree returns a tree of class phylo, with each node sorted such that the larger clade is first.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

RenumberTree

SplitFrequency *Frequency of splits*

Description

SplitFrequency provides a simple way to count the number of times that bipartition splits, as defined by a reference tree, occur in a forest of trees.

Usage

```
SplitFrequency(reference, forest)

SplitNumber(tips, tree, tipIndex, powersOf2)

ForestSplits(forest, powersOf2)

TreeSplits(tree)
```

Arguments

reference	A tree of class phylo, a Splits object.
forest	a list of trees of class phylo, or a multiPhylo object; or a Splits object.
tips	Integer vector specifying the tips of the tree within the chosen split
tree	A tree of class phylo.
tipIndex	Character vector of tip names, in a fixed order
powersOf2	Integer vector of same length as tipIndex, specifying a power of 2 to be associated with each tip in turn

Details

If multiple calculations are required, some time can be saved by using the constituent functions (see examples)

Value

Number of trees in forest that contain each split in reference. if reference is a tree of class phylo, then the sequence will correspond to the order of nodes (use 'ape::nodelabels to view). Note that the three nodes at the root of the tree correspond to a single split; see the example for how these might be plotted on a tree.

Functions

- SplitNumber: Assign a unique integer to each split
- ForestSplits: Frequency of splits in a given forest of trees
- TreeSplits: Deprecated. Listed the splits in a given tree. Use as.Splits instead.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

Examples

```
forest <- as.phylo(c(1, 10, 10, 100, 1000), nTip = 7)

# Simple, but means counting each split in the forest twice:
tree1Freqs <- SplitFrequency(forest[[1]], forest)
SplitFrequency(forest[[2]], forest)

plot(forest[[1]])
ape::nodelabels(tree1Freqs, node=as.integer(names(tree1Freqs)))
```

SplitInformation *Information content of a split*

Description

`SplitInformation` calculates the information content of a split, based on the entropy of the subset of trees consistent with the split; a split that is consistent with a smaller number of trees will have a higher information content.

Usage

```
SplitInformation(A, B)

MultiSplitInformation(partitionSizes)
```

Arguments

`A`	Number of taxa in each partition.
`B`	Number of taxa in each partition.
`partitionSizes`	Integer vector specifying the number of taxa in each partition of a multi-partition split.

Value

Information content of the split, in bits.

Functions

- `MultiSplitInformation`: Information content of a multi-partition split.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

Other split information functions: CharacterInformation, TreesMatchingSplit, UnrootedTreesMatchingSplit

Examples

```
# Eight tips can be split evenly:
SplitInformation (4, 4)

# or unevenly, which is less informative:
SplitInformation (2, 6)
```

SplitMatchProbability *Probability of matching this well*

Description

Calculates the probability that two random splits of the sizes provided will be at least as similar as the two specified.

Usage

```
SplitMatchProbability(split1, split2)

LnSplitMatchProbability(split1, split2)
```

Arguments

split1, split2	Logical vectors listing terminals in same order, such that each terminal is identified as a member of the ingroup (TRUE) or outgroup (FALSE) of the respective bipartition split.

Value

The proportion of permissible informative splits splitting the terminals into bipartitions of the sizes given, that match as well as split1 and split2 do.

Functions

- LnSplitMatchProbability: The natural logarithm of the probability

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

Examples

```
SplitMatchProbability(split1 = as.Splits(c(rep(TRUE, 4), rep(FALSE, 4))),
                      split2 = as.Splits(c(rep(TRUE, 3), rep(FALSE, 5))))
```

StringToPhyDat *String to phyDat*

Description

Converts a character string to a PhyDat object.

Usage

```
StringToPhyDat(string, tips, byTaxon = TRUE)

StringToPhydat(string, tips, byTaxon = TRUE)
```

Arguments

string	a string of tokens, optionally containing whitespace, with no terminating semi-colon.
tips,	a character vector corresponding to the names (in order) of each taxon in the matrix
byTaxon	= TRUE, string is one TAXON's coding at a time; FALSE: one CHARACTER's coding at a time

Value

This function returns a data matrix in phyDat format.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

phyDat

- PhyToString

Examples

```
morphy <- StringToPhyDat("-?01231230?-", c('Lion', 'Gazelle'), byTaxon=TRUE)
# encodes the following matrix:
# Lion     -?0123
# Gazelle  1230?-
```

`Subsplit` *Subset of a split on fewer taxa*

Description

Subset of a split on fewer taxa

Usage

```
Subsplit(splits, tips, keepAll = FALSE, unique = TRUE)
```

Arguments

`splits`	An object of class `Splits`.
`tips`	A vector specifying a subset of the tip labels applied to `split`.
`keepAll`	logical specifying whether to keep entries that define trivial splits (i.e. splits of zero or one tip) on the subset of tips.
`unique`	logical specifying whether to remove duplicate splits.

Value

An object of class `Splits`, defined on `tips`.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

Other split manipulation functions: `TrivialSplits`

Examples

```
splits <- as.Splits(PectinateTree(letters[1:9]))
efgh <- Subsplit(splits, tips = letters[5:8], keepAll = TRUE)
summary(efgh)

TrivialSplits(efgh)

Subsplit(splits, tips = letters[5:8], keepAll = FALSE)
```

Subtree *Extract subtree*

Description

Safely extracts a clade from a phylogenetic tree.

Usage

```
Subtree(tree, node)
```

Arguments

tree	A tree of class phylo, with internal numbering in cladewise order (use Preorder(tree) or (slower) Cladewise(tree)) .
node	The number of the node at the base of the clade to be extracted.

Details

Modified from the **ape** function extract.clade, which sometimes behaves erratically. Unlike extract.clade, this function supports the extraction of 'clades' that constitute a single tip.

Value

This function returns a tree of class phylo that represents a clade extracted from the original tree.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

Other tree manipulation: AddTip, Renumber, SingleTaxonTree

Examples

```
tree <- Preorder(BalancedTree(8))
plot(tree)
ape::nodelabels()
ape::nodelabels(13, 13, bg='yellow')

plot(Subtree(tree, 13))
```

SupportColour *Support colour*

Description

Colour value with which to display node support.

Usage

```
SupportColour(support, show1 = TRUE)

SupportColor(support, show1 = TRUE)
```

Arguments

support	A numeric vector of values in the range 0–1.
show1	Logical specifying whether to display values of 1. A transparent white will be returned if FALSE.

Value

A string containing the hexadecimal code for a colour picked from a diverging scale, or red if a value is invalid.

Examples

```
SupportColour(0:4 / 4, show1 = FALSE)
```

TipLabels *Extract tip labels*

Description

Extracts tip labels from an object. If the object is a single integer, TipLabels will return a vector t1, t2 ... tn, to match the default of ape::rtree.

Usage

```
TipLabels(x, single = TRUE)

## S3 method for class 'matrix'
TipLabels(x, single = TRUE)

## S3 method for class 'phylo'
TipLabels(x, single = TRUE)
```

```
## S3 method for class 'TreeNumber'
TipLabels(x, single = TRUE)

## S3 method for class 'Splits'
TipLabels(x, single = TRUE)

## S3 method for class 'list'
TipLabels(x, single = FALSE)

## S3 method for class 'multiPhylo'
TipLabels(x, single = FALSE)

## S3 method for class 'character'
TipLabels(x, single = TRUE)

## S3 method for class 'numeric'
TipLabels(x, single = TRUE)

## S3 method for class 'phyDat'
TipLabels(x, single = TRUE)

## Default S3 method:
TipLabels(x, single = TRUE)
```

Arguments

x	An object of a supported class (see Usage section).
single	Logical specifying whether to report the labels for the first object only (TRUE), or for each object in a list (FALSE).

Value

TipLabels returns a character vector listing the tip labels for the specified object.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

Other Splits operations: NSplits, NTip, TipsInSplits, as.Splits, in.Splits, match.Splits

TipsInSplits *Tips contained within splits*

Description

TipsInSplits specifies the number of tips that occur within each bipartition split in a Splits object.

Usage

```
TipsInSplits(splits, nTip = attr(splits, "nTip"))
```

Arguments

splits	Object of class Splits.
nTip	Number of tips in Splits object (inferred if not specified).

Value

A named vector of integers, specifying the number of tips contained within each split in splits.

See Also

Other Splits operations: NSplits, NTip, TipLabels, as.Splits, in.Splits, match.Splits

Examples

```
splits <- as.Splits(PectinateTree(8))
TipsInSplits(splits)
```

TreeIsRooted *Is tree rooted?*

Description

Faster alternative to of ape::is.rooted.

Usage

```
TreeIsRooted(tree)
```

Arguments

tree	A phylogenetic tree of class phylo.

Value

Logical specifying whether a root node is resolved.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

Examples

```
TreeIsRooted(BalancedTree(6))
```

`TreeNumber` *Unique integer indices for bifurcating tree topologies*

Description

Functions converting between phylogenetic trees and their unique decimal representation.

Usage

```
as.TreeNumber(x, ...)

## S3 method for class 'phylo'
as.TreeNumber(x, ...)

## S3 method for class 'multiPhylo'
as.TreeNumber(x, ...)

## S3 method for class 'character'
as.TreeNumber(x, nTip, tipLabels = TipLabels(nTip),
  ...)

## S3 method for class 'numeric'
as.phylo(x, nTip = attr(x, "nTip"),
  tipLabels = attr(x, "tip.label"), ...)

## S3 method for class 'TreeNumber'
as.phylo(x, nTip = attr(x, "nTip"),
  tipLabels = attr(x, "tip.label"), ...)
```

Arguments

`x`	Integer identifying the tree (see details).
`...`	Additional parameters for consistency with S3 methods (unused).
`nTip`	Integer specifying number of tips in the tree.
`tipLabels`	Character vector listing the labels assigned to each tip in a tree, perhaps obtained using TipLabels.

Details

There are `NUnrooted(n)` unrooted trees with n tips. As such, each n-tip tree can be uniquely identifyed by a non-negative integer $x <$ `NUnrooted(n)`.

This integer can be converted by a tree by treating it as a mixed-base number, with bases 1, 3, 5, 7, ... (2_n_ - 5).

Each digit of this mixed base number corresponds to a tip, and determines the location on a growing tree to which that tip should be added.

We start with a two-tip tree, and treat 0 as the origin of the tree.

```
0 ---- 1
```

We add tip 2 by breaking an edge and inserting a node (numbered 2 + nTip - 1). In this example, we'll work up to a six-tip tree; this node will be numbered 2 + 6 - 1 = 7. There is only one edge on which tip 2 can be added. Let's add node 7 and tip 2:

```
0 ---- 7 ---- 1
       |
       |
       2
```

There are now three edges on which tip 3 can be added. Our options are: Option 0: the edge leading to 1; Option 1: the edge leading to 2; Option 2: the edge leading to 7.

If we select option 1, we produce:

```
0 ---- 7 ---- 1
       |
       |
       8 ---- 2
       |
       |
       3
```

1 is now the final digit of our mixed-base number

There are five places to add tip 4: Option 0: the edge leading to 1; Option 1: the edge leading to 2; Option 2: the edge leading to 3; Option 3: the edge leading to 7; Option 4: the edge leading to 8.

If we chose option 3, then 3 would be the penultimate digit of our mixed-base number

If we chose option 0 for the next two additions, we could specify this tree with the mixed-base number 0021. We can convert this into decimal:

$0 \times (1 \times 3 \times 5 \times 9) +$

$0 \times (1 \times 3 \times 5) +$

$3 \times (1 \times 3) +$

$1 \times (1)$

= 10

Note that the hyperexponential nature of tree space means that there are > 2^30 unique 12-tip trees. As integers > 2^31 are not supported by R, numbers representing larger trees are represented internally as a vector of nine-digit integer 'chunks' and passed to the underlying C code, where they are combined into a single 64-bit integer. This allows trees with up to 42 tips to be accommodated.

Value

`as.phylo.numeric` returns a tree of class `phylo`.

`as.TreeNumber` returns an object of class `TreeNumber`, which comprises a numeric vector, whose elements represent successive nine-digit chunks of the decimal integer corresponding to the tree topology (in big endian order). The `TreeNumber` object has attributes `nTip` and `tip.labels`.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

References

Based on a concept by John Tromp, employed in Li *et al.* 1996.

Li M, Tromp J, Zhang L (1996). "Some notes on the nearest neighbour interchange distance." In Goos G, Hartmanis J, Leeuwen J, Cai J, Wong CK (eds.), *Computing and Combinatorics*, volume 1090, 343–351. Springer, Berlin, Heidelberg. ISBN 978-3-540-61332-9 978-3-540-68461-9, doi: 10.1007/3540613323_168.

See Also

`TreeShape`

Other tree generation functions: `BalancedTree`, `NJTree`, `PectinateTree`, `RandomTree`, `SingleTaxonTree`

Examples

```
tree <- as.phylo(10, nTip = 6)
plot(tree)
as.TreeNumber(tree)

# Larger trees:
as.TreeNumber(BalancedTree(19))

# If > 9 digits, represent the tree number as a string.
treeNumber <- as.TreeNumber("1234567890123", nTip = 14)
tree <- as.phylo(treeNumber)

as.phylo(0:2, nTip = 6, tipLabels = letters[1:6])
```

`TreesMatchingSplit` *Number of trees matching a bipartition split*

Description

Calculates the number of unrooted bifurcated trees that are consistent with a bipartition split that divides taxa into groups of size `A` and `B`.

Usage

```
TreesMatchingSplit(A, B)

LnTreesMatchingSplit(A, B)
```

Arguments

`A, B`	Number of taxa in each partition.

Value

`TreesMatchingSplit` returns a numeric specifying the number of trees that are compatible with the given split.

`LnTreesMatchingSplit` gives the natural logarithm of this number.

Functions

- `LnTreesMatchingSplit`: Logarithm of the number of trees matching a split.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

Other split information functions: `CharacterInformation`, `SplitInformation`, `UnrootedTreesMatchingSplit`

Examples

```
TreesMatchingSplit(5, 6)
LnTreesMatchingSplit(5, 6)
```

`TrivialSplits` *Are splits trivial?*

Description

Are splits trivial?

Usage

```
TrivialSplits(splits, nTip = attr(splits, "nTip"))

WithoutTrivialSplits(splits, nTip = attr(splits, "nTip"))
```

Arguments

`splits`	An object of class `Splits`.
`nTip`	Integer specifying number of tips (leaves).

Value

Logical vector specifying whether each split in `splits` is trivial, i.e. includes or excludes only a single tip or no tips at all.

Functions

- `WithoutTrivialSplits`: Remove trivial splits from a splits object

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

Other split manipulation functions: `Subsplit`

Examples

```
splits <- as.Splits(PectinateTree(letters[1:9]))
efgh <- Subsplit(splits, tips = letters[5:8], keepAll = TRUE)
summary(efgh)

TrivialSplits(efgh)
```

`UnrootedTreesMatchingSplit` *Number of trees consistent with split*

Description

Calculates the number of unrooted bifurcating trees consistent with the specified multi-partition split, using the formula of Carter *et al*. (1990).

Usage

```
UnrootedTreesMatchingSplit(splits)
```

Arguments

`splits`	A vector of integers listing the number of tips in each of a number of tree splits (e.g. bipartitions). For example, `c(3,5)` states that a character divides a set of eight tips into a group of three and a group of five.

Value

`UnrootedTreesMatchingSplit` returns an integer specifying the number of unrooted bifurcating trees consistent with the specified split.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

References

See Theorem 2 in Carter M, Hendy M, Penny D, SzÃ©kely LA, Wormald NC (1990). "On the distribution of lengths of evolutionary trees." *SIAM Journal on Discrete Mathematics*, **3**(1), 38–47. doi: 10.1137/0403005, http://epubs.siam.org/doi/abs/10.1137/0403005.

See Also

Other split information functions: `CharacterInformation`, `SplitInformation`, `TreesMatchingSplit`

Examples

```
UnrootedTreesMatchingSplit(c(3, 5))
UnrootedTreesMatchingSplit(c(3, 2, 1, 2))
```

UnshiftTree *Add Tree to Start of List*

Description

UnshiftTree adds a phylogenetic tree to the start of a list of trees. This is useful where the class of a list of trees is unknown. Adding a tree to a multiPhylo object whose own attributes apply to all trees, for example trees read from a Nexus file, causes data to be lost.

Usage

```
UnshiftTree(add, treeList)
```

Arguments

add	Tree to add to the list, of class phylo.
treeList	A list of trees, of class list, multiPhylo, or, if a single tree, phylo.

Value

A list of class list or multiPhylo (following the original class of treeList), whose first element is the tree specified as add.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

Index