

Package ‘TreeSummarizedExperiment’

December 5, 2025

Type Package

Title TreeSummarizedExperiment: a S4 Class for Data with Tree Structures

Version 2.19.0

Date 2025-03-10

Description TreeSummarizedExperiment has extended SingleCellExperiment to include hierarchical information on the rows or columns of the rectangular data.

Depends R(>= 3.6.0), SingleCellExperiment, S4Vectors (>= 0.23.18), Biostrings

License GPL (>=2)

Encoding UTF-8

LazyData true

biocViews DataRepresentation, Infrastructure

Imports methods, BiocGenerics, utils, ape, rlang, dplyr, SummarizedExperiment, BiocParallel, IRanges, treeio

VignetteBuilder knitr

Suggests ggtree, ggplot2, BiocStyle, knitr, rmarkdown, testthat

RoxygenNote 7.1.1

Collate 'TreeSummarizedExperiment.R' 'aboutLoop.R' 'allClass.R' 'aggTSE.R' 'allGenerics.R' 'changeTree.R' 'classAccessor.R' 'classValid.R' 'coercion.R' 'combine.R' 'data.R' 'deprecate_Fun.R' 'internal_utils.R' 'makeTSE.R' 'tree_addLabel.R' 'tree_asLeaf.R' 'tree_asPhylo.R' 'tree_convertNode.R' 'tree_countLeaf.R' 'tree_countNode.R' 'tree_distNode.R' 'tree_findAncestor.R' 'tree_findChild.R' 'tree_findDescendant.R' 'tree_findSibling.R' 'tree_isLeaf.R' 'tree_joinNode.R' 'tree_matTree.R' 'tree_printNode.R' 'tree_shareNode.R' 'tree_showNode.R' 'tree_toTree.R' 'tree_trackNode.R' 'tree_unionLeaf.R' 'updateObject.R'

git_url <https://git.bioconductor.org/packages/TreeSummarizedExperiment>

git_branch devel
git_last_commit d592bc4
git_last_commit_date 2025-10-29
Repository Bioconductor 3.23
Date/Publication 2025-12-04
Author Ruizhu Huang [aut, cre] (ORCID:
 <<https://orcid.org/0000-0003-3285-1945>>),
 Felix G.M. Ernst [ctb] (ORCID: <<https://orcid.org/0000-0001-5064-0928>>),
 Mark Robinson [ctb] (ORCID: <<https://orcid.org/0000-0002-3048-5518>>),
 Tuomas Borman [ctb] (ORCID: <<https://orcid.org/0000-0002-8563-8884>>)
Maintainer Ruizhu Huang <ruizhuRH@gmail.com>

Contents

TreeSummarizedExperiment-package	3
.all_equal_in_list	4
.all_have_DNAStringSet	4
.all_have_DNAStringSetList	4
.all_nonnull_in_list	5
.all_null_in_list	5
.any_null_in_list	5
.auto_rename_list	5
.bind_link_tree	6
.is_equal_link	6
.match_phylo	6
.match_phylo_list	6
.match_x_dupY	7
.name_y_with_x	7
.numeric_ij	8
.rbind_refSeq	8
.replace_link_tree_1d	9
.subset_leaf	9
.update_whichTree	10
addLabel	10
aggTSE	11
aggValue	13
asLeaf	15
asPhylo	16
changeTree	17
countLeaf	19
countNode	20
detectLoop	21
distNode	22
findAncestor	23
findChild	24
findOS	25

findSibling	26
isLeaf	27
LinkDataFrame-class	28
LinkDataFrame-constructor	28
makeTSE	29
matTree	30
phylo	31
printNode	31
rbind,TreeSummarizedExperiment-method	32
resolveLoop	33
rowLinks	34
shareNode	39
showNode	40
signalNode	41
tinyTree	42
toTree	43
trackNode	45
transNode	46
TreeSummarizedExperiment-class	47
TreeSummarizedExperiment-constructor	48
TreeSummarizedExperiment-internal	50
unionLeaf	50
updateObject,TreeSummarizedExperiment-method	51
Index	52

TreeSummarizedExperiment-package

The TreeSummarizedExperiment package

Description

TreeSummarizedExperiment implement a class of the same name, which extends SingleCellExperiment to include hierarchical information on the rows or columns of the rectangular data.

Details

It also includes an additional slot for storing reference sequences per feature.

See Also

[TreeSummarizedExperiment](#) class

`.all_equal_in_list` *test all elements in a list are equal*

Description

test all elements in a list are equal

Usage

`.all_equal_in_list(x)`

Examples

```
## Not run:
l1 <- list(a = 1, b = 2, c = 3)
l2 <- list(a = 1, b = 1, c = 1)
.all_equal_in_list(l1)
.all_equal_in_list(l2)

## End(Not run)
```

`.all_have_DNAStringSet`
test all TSEs have DNAStringSet in the referenceSeq slot

Description

test all TSEs have DNAStringSet in the referenceSeq slot

Usage

`.all_have_DNAStringSet(args)`

`.all_have_DNAStringSetList`
test all TSEs have DNAStringSetList in the referenceSeq slot

Description

test all TSEs have DNAStringSetList in the referenceSeq slot

Usage

`.all_have_DNAStringSetList(args)`

`.all_nonnull_in_list` *all elements in the list are NULL*

Description

all elements in the list are NULL

Usage

`.all_nonnull_in_list(x)`

`.all_null_in_list` *all elements in the list are NULL*

Description

all elements in the list are NULL

Usage

`.all_null_in_list(x)`

`.any_null_in_list` *Any element in the list is NULL*

Description

Any element in the list is NULL

Usage

`.any_null_in_list(x)`

`.auto_rename_list` *rename a list automatically to avoid duplicated names*

Description

rename a list automatically to avoid duplicated names

Usage

`.auto_rename_list(x)`

<code>.bind_link_tree</code>	<i>bind links & trees when combine TSE</i>
------------------------------	--

Description

bind links & trees when combine TSE

Usage

```
.bind_link_tree(x, args, drop.rowLinks, drop.colLinks, bind = "cbind")
```

<code>.is_equal_link</code>	<i>The links & trees in the specified dim are consistent</i>
-----------------------------	--

Description

The links & trees in the specified dim are consistent

Usage

```
.is_equal_link(args, dim = "row")
```

<code>.match_phylo</code>	<i>match a phylo to a list of phylo</i>
---------------------------	---

Description

match a phylo to a list of phylo

Usage

```
.match_phylo(phy, phys)
```

<code>.match_phylo_list</code>	<i>match a list of phylo (x.phys) against to a list of phylo (y.phys)</i>
--------------------------------	---

Description

match a list of phylo (x.phys) against to a list of phylo (y.phys)

Usage

```
.match_phylo_list(x.phys, y.phys)
```

<code>.match_x_dupY</code>	<i>convert char. indicator to num. indicator</i>
----------------------------	--

Description

This differs to `match` with that the duplicated values in `dy` are not ignored.

Usage

```
.match_x_dupY(x, dy)
```

Arguments

<code>x</code>	A vector. The values to be matched.
<code>dy</code>	A vector. The values to be matched against.

Author(s)

Ruizhu Huang

<code>.name_y_with_x</code>	<i>name y with x</i>
-----------------------------	----------------------

Description

name y with x

Usage

```
.name_y_with_x(x, y)
```

Examples

```
## Not run:  
x <- letters[1:5]  
y <- 1:5  
.name_y_with_x(x, y)  
  
## End(Not run)
```

<code>.numeric_ij</code>	<i>convert char. indicator to num. indicator</i>
--------------------------	--

Description

convert char. indicator to num. indicator

Usage

```
.numeric_ij(ij, x, dim = "row")
```

Arguments

<code>ij</code>	A character or numeric indicator on rows/columns of <code>x</code>
<code>x</code>	It provides row/col names for <code>ij</code> to be matched against.
<code>dim</code>	"row" or "col" to specify row/col names of <code>x</code> to be matched against.

Author(s)

Ruizhu Huang

<code>.rbind_refSeq</code>	<i>rbind referenceSeq</i>
----------------------------	---------------------------

Description

rbind referenceSeq

Usage

```
.rbind_refSeq(args)
```

`.replace_link_tree_1d` *replace row/col links & trees*

Description

replace row/col links & trees

Usage

```
.replace_link_tree_1d(x, value, ij, dim = "row")
```

Arguments

<code>x</code>	A TSE with <code>ij</code> rows/cols to be replaced by <code>value</code>
<code>value</code>	A TSE to replace some rows/cols of <code>x</code> .
<code>ij</code>	A character or numeric vector to specify which rows/cols to be replaced.
<code>dim</code>	"row" or "col" to specify the dimension is in rows or columns

Author(s)

Ruizhu Huang

`.subset_leaf` *update dimLinks and dimTree (used in subsetByLeaf)*

Description

update dimLinks and dimTree (used in subsetByLeaf)

Usage

```
.subset_leaf(x, leaf, dim = "row", updateTree = TRUE)
```

Author(s)

Ruizhu Huang

<code>.update_whichTree</code>	<i>update the 'whichTree' column in row/column link data</i>
--------------------------------	--

Description

update the 'whichTree' column in row/column link data

Usage

```
.update_whichTree(x, y)
```

Examples

```
## Not run:
(ld <- LinkDataFrame(nodeLab = letters[1:5],
                     nodeLab_alias = LETTERS[1:5],
                     nodeNum = 1:5,
                     isLeaf = TRUE,
                     whichTree = LETTERS[1:5],
                     right = 1:5))
newWhich <- setNames(letters[1:5], LETTERS[1:5])
.update_whichTree(ld, y = newWhich)

## End(Not run)
```

<code>addLabel</code>	<i>add labels to nodes of a tree</i>
-----------------------	--------------------------------------

Description

`addLabel` label nodes of a tree (phylo object)

Usage

```
addLabel(tree, label = NULL, on = c("all", "leaf", "internal"))
```

Arguments

<code>tree</code>	A phylo object
<code>label</code>	A character vector to provide node labels. The label is passed to nodes that are sorted by their node number in ascending order. The default is NULL, nodes are labeled by adding a prefix <code>Node_</code> to their node number.
<code>on</code>	Chosen from "all", "leaf", "internal". If "all", all nodes are labeled; if "leaf", leaves are labeled; if "internal", internal nodes are labeled.

Value

a phylo object

Author(s)

Ruizhu Huang

Examples

```
data(tinyTree)
library(ggtree)

# PLOT tree
# The node labels are in orange texts and the node numbers are in blue
ggtree(tinyTree, branch.length = 'none')+
  geom_text2(aes(label = label), color = "darkorange",
             hjust = -0.1, vjust = -0.7) +
  geom_text2(aes(label = node), color = "darkblue",
             hjust = -0.5, vjust = 0.7)

# change labels
nodes <- showNode(tree = tinyTree, only.leaf = FALSE)
tt <- addLabel(tree = tinyTree, label = LETTERS[nodes],
              on = "all")

ggtree(tt, branch.length = 'none')+
  geom_text2(aes(label = label), color = "darkorange",
             hjust = -0.1, vjust = -0.7) +
  geom_text2(aes(label = node), color = "darkblue",
             hjust = -0.5, vjust = 0.7)
```

aggTSE

Perform data aggregations based on the available tree structures

Description

aggTSE aggregates values on the leaf nodes of a tree to a specific arbitrary level of the tree. The level is specified via the nodes of the tree. Users could decide on which dimension (row or column) and how should the aggregation be performed.

Usage

```
aggTSE(
  x,
  rowLevel = NULL,
  rowBlock = NULL,
  colLevel = NULL,
  colBlock = NULL,
```

```

    rowFun = sum,
    colFun = sum,
    whichRowTree = 1,
    whichColTree = 1,
    whichAssay = NULL,
    message = FALSE,
    rowDataCols,
    colDataCols,
    rowFirst = TRUE,
    BPPARAM = NULL
  )

```

Arguments

<code>x</code>	A <code>TreeSummarizedExperiment</code> object.
<code>rowLevel</code>	A numeric (node numbers) or character (node labels) vector. It provides the level on the tree that data is aggregated to. The aggregation is on the row dimension. The default is <code>rowLevel = NULL</code> , and no aggregation is performed.
<code>rowBlock</code>	A column name in the <code>rowData</code> to separate the aggregation.
<code>colLevel</code>	A numeric (node numbers) or character (node labels) vector. It provides the level on the tree that data is aggregated to. The aggregation is on the column dimension. The default is <code>colLevel = NULL</code> , and no aggregation is performed.
<code>colBlock</code>	A column name in the <code>colData</code> to separate the aggregation.
<code>rowFun</code>	A function to be applied on the row aggregation. It's similar to the <code>FUN</code> in apply .
<code>colFun</code>	A function to be applied on the col aggregation. It's similar to the <code>FUN</code> in apply .
<code>whichRowTree</code>	A integer scalar or string indicating which row tree is used in the aggregation. The first row tree is used as default.
<code>whichColTree</code>	A integer scalar or string indicating which row tree is used in the aggregation. The first row tree is used as default.
<code>whichAssay</code>	A integer scalar or string indicating which assay of <code>x</code> to use in the aggregation. If <code>NULL</code> , all assay tables are used in aggregation.
<code>message</code>	A logical value. The default is <code>TRUE</code> . If <code>TRUE</code> , it will print out the running process.
<code>rowDataCols</code>	The <code>rowData</code> columns to include.
<code>colDataCols</code>	The <code>colData</code> columns to include.
<code>rowFirst</code>	<code>TRUE</code> or <code>FALSE</code> . If the aggregation is in both dims., it is performed firstly on the row dim for <code>rowFirst = TRUE</code> or on the column dim for <code>rowFirst = FALSE</code> .
<code>BPPARAM</code>	Default is <code>NULL</code> and the computation isn't run in parallel. To run computation parallelly, an optional BiocParallelParam instance determining the parallel back-end to be used during evaluation, or a list of <code>BiocParallelParam</code> instances, to be applied in sequence for nested calls to BiocParallel functions.

Value

A [TreeSummarizedExperiment](#) object

Author(s)

Ruizhu HUANG

Examples

```

# assays data
set.seed(1)
toyTable <- matrix(rnbinom(20, size = 1, mu = 10), nrow = 5)
colnames(toyTable) <- paste(rep(LETTERS[1:2], each = 2),
                           rep(1:2, 2), sep = "_")
rownames(toyTable) <- paste("entity", seq_len(5), sep = "")

toyTable

# the column data
colInf <- DataFrame(gg = c(1, 2, 3, 3),
                   group = rep(LETTERS[1:2], each = 2),
                   row.names = colnames(toyTable))

colInf

# the toy tree
library(ape)
set.seed(4)
treeC <- rtree(4)
treeC$node.label <- c("All", "GroupA", "GroupB")

library(ggtree)
ggtree(treeC, size = 2) +
  geom_text2(aes(label = node), color = "darkblue",
            hjust = -0.5, vjust = 0.7, size = 6) +
  geom_text2(aes(label = label), color = "darkorange",
            hjust = -0.1, vjust = -0.7, size = 6)

tse <- TreeSummarizedExperiment(assays = list(toyTable),
                              colData = colInf,
                              colTree = treeC,
                              colNodeLab = treeC$tip.label,
                              metadata = list(test = 1:4))

aggCol <- aggTSE(x = tse, colLevel = c("GroupA", "GroupB"),
               colFun = sum)

assays(aggCol)[[1]]

```

Description

aggValue aggregates values on the leaf nodes of a tree to a specific arbitrary level of the tree. The level is specified via the nodes of the tree. Users could decide on which dimension (row or column) and how should the aggregation be performed.

Usage

```
aggValue(
  x,
  rowLevel = NULL,
  rowBlock = NULL,
  colLevel = NULL,
  colBlock = NULL,
  FUN = sum,
  assay = NULL,
  message = FALSE
)
```

Arguments

x	A TreeSummarizedExperiment object.
rowLevel	A numeric (node numbers) or character (node labels) vector. It provides the level on the tree that data is aggregated to. The aggregation is on the row dimension. The default is rowLevel = NULL, and no aggregation is performed.
rowBlock	A column name in the rowData to separate the aggregation.
colLevel	A numeric (node numbers) or character (node labels) vector. It provides the level on the tree that data is aggregated to. The aggregation is on the column dimension. The default is colLevel = NULL, and no aggregation is performed.
colBlock	A column name in the colData to separate the aggregation.
FUN	A function to be applied on the aggregation. It's similar to the FUN in apply .
assay	A integer scalar or string indicating which assay of x to use in the aggregation. If NULL, all assay tables are used in aggregation.
message	A logical value. The default is TRUE. If TRUE, it will print out the running process.

Value

A TreeSummarizedExperiment object or a matrix. The output has the same class of the input x.

Author(s)

Ruizhu HUANG

See Also

[aggTSE](#)

asLeaf	<i>change internal nodes to leaf nodes</i>
--------	--

Description

asLeaf updates a phylo tree by changing the specified internal nodes to leaf nodes. In other words, the descendant nodes of the specified internal nodes are removed.

Usage

```
asLeaf(tree, node)
```

Arguments

tree	A phylo object.
node	A numeric or character vector. It specifies internal nodes that are changed to leaves via their node labels or numbers.

Value

A phylo object.

Examples

```
library(ggtree)
data(tinyTree)
ggtree(tinyTree, ladderize = FALSE) +
  geom_text2(aes(label = label), color = "darkorange",
             hjust = -0.1, vjust = -0.7) +
  geom_text2(aes(label = node), color = "darkblue",
             hjust = -0.5, vjust = 0.7) +
  geom_highlight(node = 16) +
  geom_point2()

# remove the blue branch
NT1 <- asLeaf(tree = tinyTree, node = 16)

ggtree(NT1, ladderize = FALSE) +
  geom_text2(aes(label = label), color = "darkorange",
             hjust = -0.1, vjust = -0.7) +
  geom_point2()

# if mergeSingle = TRUE, the node (Node_17) is removed.
NT2 <- asLeaf(tree = tinyTree, node = c(15, 13))

ggtree(NT2, ladderize = FALSE) +
  geom_text2(aes(label = label), color = "darkorange",
             hjust = -0.1, vjust = -0.7) +
```

```
geom_point2()
```

asPhylo

Convert a data frame to a phylo object

Description

asPhylo converts a data frame to a phylo object. Compared to toTree, asPhylo allows the output tree to have different number of nodes in paths connecting leaves to the root.

Usage

```
asPhylo(data, column_order = NULL, asNA = NULL)
```

Arguments

data	A data frame or matrix.
column_order	A vector that includes the column names of data to reorder columns of data. Default is NULL, the original order of data is kept.
asNA	This specifies strings that are considered as NA

Details

The last column is used as the leaf nodes

Value

a phylo object

Author(s)

Ruizhu Huang

Examples

```
library(ggtree)

# Example 0:
taxTab <- data.frame(R1 = rep("A", 5),
                     R2 = c("B1", rep("B2", 4)),
                     R3 = paste0("C", 1:5))
# Internal nodes: their labels are prefixed with colnames of taxTab
# e.g., R2:B2
taxTree <- asPhylo(data = taxTab)
ggtree(taxTree) +
  geom_text2(aes(label = label), color = "red", vjust = 1) +
  geom_nodepoint()
```

```

# (Below gives the same output as toTree)
taxTab$R1 <- paste0("R1:", taxTab$R1)
taxTab$R2 <- paste0("R2:", taxTab$R2)
taxTree <- asPhylo(data = taxTab)
# viz the tree
ggtree(taxTree) +
  geom_text2(aes(label = label), color = "red", vjust = 1) +
  geom_nodepoint()

# Example 1
df1 <- rbind.data.frame(c("root", "A1", "A2", NA),
                        c("root", "B1", NA, NA))
colnames(df1) <- paste0("L", 1:4)
tree1 <- asPhylo(df1)

ggtree(tree1, color = "grey") +
  geom_nodepoint() +
  geom_text2(aes(label = label), angle = 90,
             color = "red", vjust = 2,
             size = 4)

# Example 2
df2 <- data.frame(Group_1 = rep("Root", 11),
                  Group_2 = rep(c(13, 21), c(9, 2)),
                  Group_3 = rep(c(14, 18, "unknown"), c(5, 4, 2)),
                  Group_4 = rep(c(15, "unknown", 19, "unknown"), c(4, 1, 3, 3)),
                  Group_5 = rep(c(16, "unknown", 20, "unknown"), c(3, 2, 2, 4)),
                  Group_6 = rep(c(17, "unknown"), c(2, 9)),
                  LEAF = 1:11)

tree2 <- asPhylo(df2, asNA = "unknown")

ggtree(tree2, color = "grey") +
  geom_nodepoint() +
  geom_text2(aes(label = label), angle = 90,
             color = "red", vjust = 2,
             size = 4)

# Example 3
df3 <- df2
df3[10:11, 3] <- ""

tree3 <- asPhylo(df3, asNA = c("unknown", ""))

ggtree(tree3, color = "grey") +
  geom_nodepoint() +
  geom_text2(aes(label = label), angle = 90,
             color = "red", vjust = 2,
             size = 4)

```

Description

changeTree changes a row or column tree in a TreeSummarizedExperiment object.

Usage

```
changeTree(
  x,
  rowTree = NULL,
  rowNodeLab = NULL,
  colTree = NULL,
  colNodeLab = NULL,
  whichRowTree = 1,
  whichColTree = 1
)
```

Arguments

x	A TreeSummarizedExperiment object
rowTree	A phylo object. A new row tree.
rowNodeLab	A character string. It provides the labels of nodes that the rows of assays tables corresponding to. If NULL (default), the row names of the assays tables are used.
colTree	A phylo object. A new column tree.
colNodeLab	A character string. It provides the labels of nodes that the columns of assays tables corresponding to. If NULL (default), the column names of the assays tables are used.
whichRowTree	Which row tree to be replaced? Default is 1 (the first tree in the rowTree slot).
whichColTree	Which column tree to be replaced? Default is 1 (the first tree in the colTree slot).

Value

A TreeSummarizedExperiment object

Author(s)

Ruizhu Huang

Examples

```
library(ape)
set.seed(1)
treeR <- ape::rtree(10)

# the count table
count <- matrix(rpois(160, 50), nrow = 20)
rownames(count) <- paste0("entity", 1:20)
colnames(count) <- paste("sample", 1:8, sep = "_")
```

```

# The sample information
sampC <- data.frame(condition = rep(c("control", "trt"),
                                each = 4),
                    gender = sample(x = 1:2, size = 8,
                                replace = TRUE))

rownames(sampC) <- colnames(count)
# build a TreeSummarizedExperiment object
tse <- TreeSummarizedExperiment(assays = list(count),
                                colData = sampC,
                                rowTree = treeR,
                                rowNodeLab = rep(treeR$tip.label, each = 2))

treeR2 <- drop.tip(phy = treeR, tip = c("t10", "t9", "t8"))

# if rownames are not used in node labels of the tree, provide rowNodeLab
use <- changeTree(x = tse, rowTree = treeR2,
                  rowNodeLab = rep(treeR$tip.label, each = 2))

use

# if rownames are used in node labels of tree, rowNodeLab is not required.

rownames(tse) <- rep(treeR$tip.label, each = 2)
cse <- changeTree(x = tse, rowTree = treeR2)
cse

```

countLeaf	<i>count the number of leaf nodes</i>
-----------	---------------------------------------

Description

countLeaf calculates the number of leaves on a phylo tree.

Usage

```
countLeaf(tree)
```

Arguments

tree	A phylo object
------	----------------

Value

a numeric value

Author(s)

Ruizhu Huang

Examples

```
library(ggtree)

data(tinyTree)

ggtree(tinyTree, branch.length = 'none') +
  geom_text2(aes(label = label), hjust = -0.3) +
  geom_text2(aes(label = node), vjust = -0.8,
    hjust = -0.3, color = 'blue')

(n <- countLeaf(tinyTree))
```

countNode	<i>count the number of nodes</i>
-----------	----------------------------------

Description

countNode calculates the number of nodes on a phylo tree.

Usage

```
countNode(tree)
```

Arguments

tree	A phylo object
------	----------------

Value

a numeric value

Author(s)

Ruizhu Huang

Examples

```
library(ggtree)

data(tinyTree)

ggtree(tinyTree, branch.length = 'none') +
  geom_text2(aes(label = label), hjust = -0.3) +
  geom_text2(aes(label = node), vjust = -0.8,
    hjust = -0.3, color = 'blue')
```

```
(n <- countNode(tinyTree))
```

detectLoop	<i>Detect loops detectLoop detects loops</i>
------------	--

Description

Detect loops detectLoop detects loops

Usage

```
detectLoop(tax_tab)
```

Arguments

tax_tab a data frame where columns store hierarchical levels. The columns from the left to the right correspond nodes from the root to the leaf.

Value

a data frame

Author(s)

Ruizhu Huang

Examples

```
df <- data.frame(A = rep("a", 8),
                  B = rep(c("b1", "b2", "b3", "b4"), each = 2),
                  C = paste0("c", c(1, 2, 2, 3:7)),
                  D = paste0("d", 1:8))

# The result means that a loop is caused by 'b1' and 'b2' in column 'B' and
# 'c2' in column 'C' (a-b1-c2; a-b2-c2)
detectLoop(tax_tab = df)

df <- data.frame(R1 = rep("A", 6),
                  R2 = c("B1", rep("B2", 4), "B3"),
                  R3 = c("C1", "C2", "C3", NA, NA, NA),
                  R4 = c("D1", "D2", "D3", NA, NA, NA),
                  R5 = paste0("E", 1:6))
detectLoop(tax_tab = df)

df <- data.frame(R1 = rep("A", 7),
                  R2 = c("B1", rep("B2", 4), "B3", "B3"),
                  R3 = c("C1", "C2", "C3", "", "", "", ""),
                  R4 = c("D1", "D2", "D3", "", "", "", ""),
                  R5 = paste0("E", 1:7))
```

```

detectLoop(tax_tab = df)

df <- data.frame(R1 = rep("A", 7),
                 R2 = c("B1", rep("B2", 4), "B3", "B3"),
                 R3 = c("C1", "C2", "C3", NA, NA, NA, NA),
                 R4 = c("D1", "D2", "D3", NA, NA, NA, NA),
                 R5 = paste0("E", 1:7))
detectLoop(tax_tab = df)

```

distNode

Calculate the distance between any two nodes on the tree

Description

distNode is to calculate the distance between any two nodes on a phylo tree

Usage

```
distNode(tree, node)
```

Arguments

tree	A phylo object.
node	A numeric or character vector of length two.

Value

A numeric value.

Examples

```

library(ggtree)
data(tinyTree)
ggtree(tinyTree) +
  geom_text2(aes(label = node), color = "darkorange",
             hjust = -0.1, vjust = -0.7) +
  geom_text2(aes(label = branch.length), color = "darkblue",
             vjust = 0.7)

distNode(tree = tinyTree, node = c(10, 11))
distNode(tree = tinyTree, node = c(12, 13))
distNode(tree = tinyTree, node = c(13, 15))
distNode(tree = tinyTree, node = c(12, 14))

```

findAncestor	<i>Find the ancestors of specified nodes</i>
--------------	--

Description

findAncestor finds the ancestor in the nth generation above specified nodes.

Usage

```
findAncestor(tree, node, level, use.alias = FALSE)
```

Arguments

tree	A phylo object
node	A vector of node numbers or node labels
level	A vector of numbers to define nth generation before the specified nodes
use.alias	A logical value, TRUE or FALSE. The default is FALSE, and the node label would be used to name the output; otherwise, the alias of node label would be used to name the output. The alias of node label is created by adding a prefix "alias_" to the node number.

Value

A vector of nodes. The numeric value is the node number, and the vector name is the corresponding node label. If a node has no label, it would have NA as name when use.alias = FALSE, and have the alias of node label as name when use.alias = TRUE.

Author(s)

Ruizhu Huang

Examples

```
library(ggtree)
data(tinyTree)
ggtree(tinyTree, branch.length = 'none') +
  geom_text2(aes(label = label), color = "darkorange",
             hjust = -0.1, vjust = -0.7) +
  geom_text2(aes(label = node), color = "darkblue",
             hjust = -0.5, vjust = 0.7)

findAncestor(tree = tinyTree, node = c(18, 13), level = 1)
```

findChild

Find the children

Description

findChild finds children of an internal node.

Usage

```
findChild(tree, node = 11, use.alias = FALSE)
```

Arguments

tree	A phylo object.
node	An internal node. It could be the node number or the node label.
use.alias	A logical value, TRUE or FALSE. The default is FALSE, and the node label would be used to name the output; otherwise, the alias of node label would be used to name the output. The alias of node label is created by adding a prefix "alias_" to the node number.

Value

A vector of nodes. The numeric value is the node number, and the vector name is the corresponding node label. If a node has no label, it would have NA as name when use.alias = FALSE, and have the alias of node label as name when use.alias = TRUE.

Author(s)

Ruizhu Huang

Examples

```
data(tinyTree)

library(ggtree)
ggtree(tinyTree) +
  geom_text2(aes(label = node), color = "darkblue",
             hjust = -0.5, vjust = 0.7) +
  geom_highlight(node = 17, fill = 'steelblue', alpha = 0.5) +
  geom_text2(aes(label = label), color = "darkorange",
             hjust = -0.1, vjust = -0.7)

(tips <- findChild(tree = tinyTree, node = 17))
```

findOS	<i>Find descendants (or offsprings)</i>
--------	---

Description

findDescendant finds descendants of a node.

Usage

```
findOS(tree, node, only.leaf = TRUE, self.include = FALSE, use.alias = FALSE)
```

```
findDescendant(
  tree,
  node,
  only.leaf = TRUE,
  self.include = FALSE,
  use.alias = FALSE
)
```

Arguments

tree	A phylo object.
node	An internal node. It could be the node number or the node label.
only.leaf	A logical value, TRUE or FALSE. The default is TRUE. If default, only the leaf nodes in the descendant nodes would be returned.
self.include	A logical value, TRUE or FALSE. The default is FALSE. If TRUE, the node specified in node is included and the leaf node itself is returned as its descendant.
use.alias	A logical value, TRUE or FALSE. The default is FALSE, and the node label would be used to name the output; otherwise, the alias of node label would be used to name the output. The alias of node label is created by adding a prefix "alias_" to the node number.

Value

A vector of nodes. The numeric value is the node number, and the vector name is the corresponding node label. If a node has no label, it would have NA as name when `use.alias = FALSE`, and have the alias of node label as name when `use.alias = TRUE`.

Author(s)

Ruizhu Huang

Examples

```

data(tinyTree)

library(ggtree)
ggtree(tinyTree) +
  geom_text2(aes(label = node), color = "darkblue",
             hjust = -0.5, vjust = 0.7) +
  geom_highlight(node = 17, fill = 'steelblue', alpha = 0.5) +
  geom_text2(aes(label = label), color = "darkorange",
             hjust = -0.1, vjust = -0.7)

(tips <- findDescendant(tree = tinyTree, node = c(17), only.leaf = TRUE))

```

findSibling	<i>find the sibling node</i>
-------------	------------------------------

Description

findSibling is to find the sibling node of an node node.

Usage

```
findSibling(tree, node, use.alias = FALSE)
```

Arguments

tree	A phylo object.
node	A numeric or character vector. Node labels or node numbers.
use.alias	A logical value, TRUE or FALSE. The default is FALSE, and the original node label would be used to name the output; otherwise, the alias of node label would be used to name the output. The alias of node label is created by adding a prefix "alias_" to the node number.

Value

A vector of nodes. The numeric value is the node number, and the vector name is the corresponding node label. If a node has no label, it would have NA as name when use.alias = FALSE, and have the alias of node label as name when use.alias = TRUE.

Examples

```

library(ggtree)
data(tinyTree)
ggtree(tinyTree, branch.length = 'none') +
  geom_text2(aes(label = label), color = "darkorange",
             hjust = -0.1, vjust = -0.7) +
  geom_text2(aes(label = node), color = "darkblue",
             hjust = -0.5, vjust = 0.7)

```

```
findSibling(tree = tinyTree, node = 17)
findSibling(tree = tinyTree, node = c(13, 17))
```

isLeaf

To test whether the specified nodes are leaf nodes

Description

isLeaf is to test wheter some specified nodes are leaf nodes of a tree.

Usage

```
isLeaf(tree, node)
```

Arguments

tree	A phylo object.
node	A numeric or character vector. Node labels or node numbers.

Value

a logical vector with the same length as the input node.

Author(s)

Ruizhu HUANG

Examples

```
data(tinyTree)
library(ggtree)

# PLOT tree
# The node labels are in orange texts and the node numbers are in blue
ggtree(tinyTree, branch.length = 'none') +
  geom_text2(aes(label = label), color = "darkorange",
    hjust = -0.1, vjust = -0.7) +
  geom_text2(aes(label = node), color = "darkblue",
    hjust = -0.5, vjust = 0.7)

isLeaf(tree = tinyTree, node = c(5, 4, 18))
isLeaf(tree = tinyTree, node = c("t4", "t9", "Node_18" ))
```

LinkDataFrame-class	<i>LinkDataFrame: A S4 class extended from DataFrame An S4 class LinkDataFrame</i>
---------------------	--

Description

The **LinkDataFrame** is extended from the class **DataFrame** to include at least four columns nodeLab, nodeLab_alias, nodeNum, and isLeaf.

Constructor

See [LinkDataFrame-constructor](#) for constructor functions.

LinkDataFrame-constructor	<i>Construct a LinkDataFrame Construct a LinkDataFrame object</i>
---------------------------	---

Description

Construct a LinkDataFrame Construct a LinkDataFrame object

Usage

```
LinkDataFrame(nodeLab, nodeLab_alias, nodeNum, isLeaf, whichTree, ...)
```

Arguments

nodeLab	A character vector
nodeLab_alias	A character vector
nodeNum	A numeric vector
isLeaf	A logical vector
whichTree	A character vector
...	All arguments accepted by DataFrame-class .

Value

A LinkDataFrame object

See Also

[LinkDataFrame](#) [DataFrame](#)

Examples

```
(ld <- LinkDataFrame(nodeLab = letters[1:5],
                    nodeLab_alias = LETTERS[1:5],
                    nodeNum = 1:5,
                    isLeaf = TRUE,
                    whichTree = LETTERS[1:5],
                    right = 1:5))
```

makeTSE

A toy TreeSummarizedExperiment object

Description

makeTSE creates a toy TreeSummarizedExperiment object.

Usage

```
makeTSE(nrow = 10, ncol = 4, include.rowTree = TRUE, include.colTree = TRUE)
```

Arguments

nrow	a numeric value to specify the number of rows of TreeSummarizedExperiment
ncol	a numeric value to specify the number of columns of TreeSummarizedExperiment
include.rowTree	TRUE or FALSE. Default is TRUE, so the output TreeSummarizedExperiment has a phylo object in rowTree.
include.colTree	TRUE or FALSE. Default is TRUE, so the output TreeSummarizedExperiment has a phylo object in colTree.

Details

The assays contains a matrix with values from 1:(nrow*ncol). The rowData has two columns, var1 and var2. var1 is created with `rep_len(letters, nrow)`. var2 is created with `rep_len(c(TRUE, FALSE), nrow)`. The colData has two columns, ID and group. ID is created with `seq_len(ncol)`. group is created with `rep_len(LETTERS[1:2], ncol)`. The row/col tree is generated with `ape::rtree()`. So, to generate reproducible trees, `set.seed()` is required.

Value

A TreeSummarizedExperiment object

Author(s)

Ruizhu Huang

Examples

```
set.seed(1)
makeTSE()
```

matTree	<i>Transform a phylo object into a matrix.</i>
---------	--

Description

matTree transforms a phylo tree into a matrix. The entry of the matrix is node number. Each row represents a path connecting a leaf node and the root. The columns are arranged in the order as the path passing the nodes to reach the root.

Usage

```
matTree(tree)
```

Arguments

tree	A phylo object
------	----------------

Value

A matrix

Author(s)

Ruizhu Huang

Examples

```
library(ggtree)

data(tinyTree)
ggtree(tinyTree, branch.length = 'none') +
  geom_text2(aes(label = node))

# each row of the matrix representing a path.
# the first column is leaf nodes; the last non-NA value in a row is the root
mat <- matTree(tree = tinyTree)
```

phylo	<i>phylo: A S3 class for the phylogenetic tree</i>
-------	--

Description

The **ape** package does not export its phylo class, probably because it is not really defined formally anywhere. Technically, it is an S3 class extended from the class list. Any exported definitions from the **ape** package would be preferred to use if available.

Usage

```
phylo
```

Format

An object of class phylo of length 0.

printNode	<i>To print out the node labels</i>
-----------	-------------------------------------

Description

nodeLabel is to print out the node labels of a phylo tree.

Usage

```
printNode(tree, type = c("leaf", "internal", "all"))
```

Arguments

tree	A phylo object.
type	A character value choose from leaf, all, and internal. If leaf, the output is a data frame including only leaf nodes; if internal, the output is a data frame including only internal nodes; if all, the output is a data frame including all nodes.

Value

a data frame

Author(s)

Ruizhu HUANG

Examples

```
data(tinyTree)
library(ggtree)

# PLOT tree
# The node labels are in orange texts and the node numbers are in blue
ggtree(tinyTree,branch.length = 'none')+
  geom_text2(aes(label = label), color = "darkorange",
             hjust = -0.1, vjust = -0.7) +
  geom_text2(aes(label = node), color = "darkblue",
             hjust = -0.5, vjust = 0.7)

(pn1 <- printNode(tinyTree, type = "leaf"))
(pn2 <- printNode(tinyTree, type = "internal"))
(pn3 <- printNode(tinyTree, type = "all"))
```

rbind,TreeSummarizedExperiment-method

Combine TSEs by rows or columns

Description

rbind and cbind take one or more TreeSummarizedExperiment objects and combine them by columns or rows, respectively.

Usage

```
## S4 method for signature 'TreeSummarizedExperiment'
rbind(..., deparse.level = 1)

## S4 method for signature 'TreeSummarizedExperiment'
cbind(..., deparse.level = 1)
```

Arguments

... One or more TreeSummarizedExperiment objects.
deparse.level See [cbind](#)

Value

A TreeSummarizedExperiment object

Author(s)

Ruizhu Huang

Examples

```

# rbind works :
# a) TSE without rowTree and without colTree
# b) TSE with rowTree but without colTree
# c) TSE without rowTree but with colTree
# d) TSE with rowTree & colTree

set.seed(1)
# a)
(tse_a <- makeTSE(include.colTree = FALSE))
(tse_b <- makeTSE(include.colTree = FALSE))

# b)
(tse_c <- makeTSE(include.rowTree = FALSE))
(tse_d <- makeTSE(include.rowTree = FALSE))

rbind(tse_a, tse_b)
cbind(tse_c, tse_d)

```

resolveLoop	<i>Resolve loops resolveLoop resolve loops by adding suffix to the child node. The suffix is "_i" where 'i' is a number. Please see examples.</i>
-------------	---

Description

Resolve loops resolveLoop resolve loops by adding suffix to the child node. The suffix is "_i" where 'i' is a number. Please see examples.

Usage

```
resolveLoop(tax_tab)
```

Arguments

tax_tab	a data frame where columns store hierarchical levels. The columns from the left to the right correspond nodes from the root to the leaf.
---------	--

Value

a data frame

Author(s)

Ruizhu Huang

Examples

```
# example 1
df <- data.frame(A = rep("a", 8),
                 B = rep(c("b1", "b2", "b3", "b4"), each = 2),
                 C = paste0("c", c(1, 2, 2, 3:7)),
                 D = paste0("d", 1:8))

# The result means that a loop is caused by 'b1' and 'b2' in column 'B' and
# 'c2' in column 'C' (a-b1-c2; a-b2-c2)
resolveLoop(tax_tab = df)

# example 2
taxTab <- data.frame(R1 = rep("A", 5),
                     R2 = c("B1", rep("B2", 3), ""),
                     R3 = c("C1", "C2", "C3", "", ""),
                     R4 = c("D1", "D2", "D3", "", ""),
                     R5 = paste0("E", 1:5))

resolveLoop(tax_tab = taxTab)

# example 3
taxTab <- data.frame(R1 = rep("A", 6),
                     R2 = c("B1", rep("B2", 4), ""),
                     R3 = c("C1", "C2", "C3", "", "", ""),
                     R4 = c("D1", "D2", "D3", "", "", ""),
                     R5 = paste0("E", 1:6))

resolveLoop(tax_tab = taxTab)

# example 3
taxTab <- data.frame(
  R1 = rep("A", 5),
  R2 = c("B1", rep("B2", 3), "B3"),
  R3 = c("C1", "C2", "C3", NA, NA),
  R4 = c("D1", "D2", "D3", NA, NA),
  R5 = paste0("E", 1:5))
resolveLoop(tax_tab = taxTab)
```

rowLinks

TreeSummarizedExperiment-accessors

Description

All accessor functions that work on [SingleCellExperiment](#) should work on **TreeSummarized-Experiment**. Additionally, new accessors rowLinks collinks, rowTree and colTree accessor function are available for **TreeSummarizedExperiment**.

Usage

```
rowLinks(x)

## S4 method for signature 'TreeSummarizedExperiment'
rowLinks(x)

colLinks(x)

## S4 method for signature 'TreeSummarizedExperiment'
colLinks(x)

rowTree(x, whichTree = 1, value)

## S4 method for signature 'TreeSummarizedExperiment'
rowTree(x, whichTree = 1, value)

rowTree(x, whichTree = 1) <- value

## S4 replacement method for signature 'TreeSummarizedExperiment'
rowTree(x, whichTree = 1) <- value

colTree(x, whichTree = 1)

## S4 method for signature 'TreeSummarizedExperiment'
colTree(x, whichTree = 1)

colTree(x, whichTree = 1) <- value

## S4 replacement method for signature 'TreeSummarizedExperiment'
colTree(x, whichTree = 1) <- value

rowTreeNames(x, value)

## S4 method for signature 'TreeSummarizedExperiment'
rowTreeNames(x, value)

rowTreeNames(x) <- value

## S4 replacement method for signature 'TreeSummarizedExperiment'
rowTreeNames(x) <- value

colTreeNames(x, value)

## S4 method for signature 'TreeSummarizedExperiment'
colTreeNames(x, value)

colTreeNames(x) <- value
```

```

## S4 replacement method for signature 'TreeSummarizedExperiment'
colTreeNames(x) <- value

referenceSeq(x)

## S4 method for signature 'TreeSummarizedExperiment'
referenceSeq(x)

referenceSeq(x) <- value

## S4 replacement method for signature 'TreeSummarizedExperiment'
referenceSeq(x) <- value

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

## S4 replacement method for signature
## 'TreeSummarizedExperiment,ANY,ANY,TreeSummarizedExperiment'
x[i, j, ...] <- value

## S4 replacement method for signature 'TreeSummarizedExperiment'
rownames(x) <- value

## S4 replacement method for signature 'TreeSummarizedExperiment'
colnames(x) <- value

subsetByLeaf(
  x,
  rowLeaf,
  colLeaf,
  whichRowTree,
  whichColTree,
  updateTree = TRUE
)

## S4 method for signature 'TreeSummarizedExperiment'
subsetByLeaf(
  x,
  rowLeaf,
  colLeaf,
  whichRowTree,
  whichColTree,
  updateTree = TRUE
)

subsetByNode(x, rowNode, colNode, whichRowTree, whichColTree)

## S4 method for signature 'TreeSummarizedExperiment'

```

```
subsetByNode(x, rowNode, colNode, whichRowTree, whichColTree)
```

Arguments

x	A TreeSummarizedExperiment object
whichTree	A numeric indicator or name character to specify which tree in the rowTree or colTree to be extracted. The default is to extract the first tree. If whichTree = NULL, a list of all trees is extracted.
value	- the new rownames or colnames as a character value. See BiocGenerics. - A DNAStrngSet object or an object coercible to one
i, j	The row, column index to subset x. The arguments of the subset function []
...	The argument from the subset function []
drop	A logical value, TRUE or FALSE. The argument from the subset function []
rowLeaf	A vector of leaves that are used to subset rows. One could use the leaf number, or the leaf label to specify nodes, but not a mixture of them.
colLeaf	A vector of leaves that are used to subset columns. One could use the leaf number, or the leaf label to specify nodes, but not a mixture of them.
whichRowTree	A numeric indicator or name character to specify which tree in the rowTree.
whichColTree	A numeric indicator or name character to specify which tree in the colTree.
updateTree	TRUE or FALSE. Default is TRUE, which updates tree structures after subsetting.
rowNode	A vector of nodes that are used to subset rows. One could use the node number, the node label or the node alias to specify nodes, but not a mixture of them.
colNode	A vector of nodes that are used to subset columns. One could use the node number, the node label or the node alias to specify nodes, but not a mixture of them.

Value

Elements from TreeSummarizedExperiment.

Author(s)

Ruizhu HUANG

See Also

[TreeSummarizedExperiment](#) [SingleCellExperiment](#)

Examples

```
# the assay table
set.seed(1)
y <- matrix(rnbinom(300,size=1,mu=10),nrow=10)
colnames(y) <- paste(rep(LETTERS[1:3], each = 10), rep(1:10,3), sep = "_")
rownames(y) <- tinyTree$tip.label
```

```

# the row data
rowInf <- DataFrame(var1 = sample(letters[1:3], 10, replace = TRUE),
                    var2 = sample(c(TRUE, FALSE), 10, replace = TRUE))
# the column data
colInf <- DataFrame(gg = factor(sample(1:3, 30, replace = TRUE)),
                    group = rep(LETTERS[1:3], each = 10))

# the tree structure on the rows of assay tables
data("tinyTree")

# the tree structure on the columns of assay tables
sampTree <- ape::rtree(30)
sampTree$tip.label <- colnames(y)

# create the TreeSummarizedExperiment object
toy_tse <- TreeSummarizedExperiment(assays = list(y),
                                   rowData = rowInf,
                                   colData = colInf,
                                   rowTree = tinyTree,
                                   colTree = sampTree)

## extract the rowData
(rowD <- rowData(x = toy_tse))

## extract the colData
(colD <- colData(x = toy_tse))

## extract the linkData
# on rows
(rowL <- rowLinks(x = toy_tse))
# on columns
(coll <- collinks(x = toy_tse))

## extract the treeData
# on rows
(rowT <- rowTree(x = toy_tse))
# on columns
(colT <- colTree(x = toy_tse))

# the referenceSeq data
refSeq <- DNASTringSetList(one = DNASTringSet(rep("A", nrow(toy_tse))),
                           two = DNASTringSet(rep("B", nrow(toy_tse))))
referenceSeq(toy_tse) <- refSeq
toy_tse

# subset treeSE by leaves
library(ape)
set.seed(1)
z <- makeTSE(nrow = 5, ncol = 4, include.rowTree = TRUE, include.colTree = FALSE)
y <- makeTSE(nrow = 4, ncol = 4, include.rowTree = TRUE, include.colTree = FALSE)
tr <- ape::rtree(4)
zy <- rbind(z, y)

```

```

x <- changeTree(x = zy, rowTree = tr, whichRowTree = 2, rowNodeLab = tr$tip.label)
rowLinks(zy)
rowLinks(x)
## 1) rowLeaf exist only in one of trees
rf <- c("t1", "t3")
sx <- subsetByLeaf(x = x, rowLeaf = rf)
rowLinks(sx)

sx <- subsetByLeaf(x = x, rowLeaf = rf, updateTree = FALSE)
rowLinks(sx)

## 2) rowLeaf exist in all trees
rf <- 1:3
sxx <- subsetByLeaf(x = x, rowLeaf = rf)
rowLinks(sxx)

## 3) rowLeaf exist in all trees, but subset and update only the specified
trees
rf <- c(3:4)
sxx <- subsetByLeaf(x = x, rowLeaf = rf, whichRowTree = "phylo")
rowLinks(sxx)

```

shareNode

Find the share node

Description

shareNode is to find the node where the specified nodes first meet.

Usage

```
shareNode(tree, node, use.alias = FALSE)
```

Arguments

tree	A phylo object.
node	A vector of node numbers or node labels.
use.alias	A logical value, TRUE or FALSE. The default is FALSE, and the node label would be used to name the output; otherwise, the alias of node label would be used to name the output. The alias of node label is created by adding a prefix "alias_" to the node number.

Value

A vector of nodes. The numeric value is the node number, and the vector name is the corresponding node label. If a node has no label, it would have NA as name when use.alias = FALSE, and have the alias of node label as name when use.alias = TRUE.

Author(s)

Ruizhu Huang

Examples

```
library(ggtree)
data(tinyTree)

# PLOT tree
ggtree(tinyTree, branch.length = 'none') +
  geom_text2(aes(label = label), color = "darkorange",
    hjust = -0.1, vjust = -0.7) +
  geom_text2(aes(label = node), color = "darkblue",
    hjust = -0.5, vjust = 0.7)

## find the node shared by provided node labels
shareNode(node = c('t4','t9'), tree = tinyTree,
  use.alias = FALSE)

shareNode(node = c('t10','Node_17'), tree = tinyTree,
  use.alias = FALSE)

## find the node shared by provided node numbers
shareNode(node = c(2, 3), tree = tinyTree)
```

showNode	<i>Find nodes on the tree</i>
----------	-------------------------------

Description

showNode is to get nodes from the tree.

Usage

```
showNode(tree, only.leaf = FALSE, use.alias = FALSE)
```

Arguments

tree	A phylo object.
only.leaf	A logical value, TRUE or FALSE. The default is FALSE, all nodes are output; otherwise, leaves are output
use.alias	A logical value, TRUE or FALSE. The default is FALSE, and the node label would be used to name the output; otherwise, the alias of node label would be used to name the output. The alias of node label is created by adding a prefix "alias_" to the node number.

Value

A vector of nodes. The numeric value is the node number, and the vector name is the corresponding node label. If a node has no label, it would have NA as name when `use.alias = FALSE`, and have the alias of node label as name when `use.alias = TRUE`.

Author(s)

Ruizhu Huang

Examples

```
library(ggtree)
data(tinyTree)

# PLOT tree
ggtree(tinyTree, branch.length = 'none') +
  geom_text2(aes(label = label), color = "darkorange",
             hjust = -0.1, vjust = -0.7) +
  geom_text2(aes(label = node), color = "darkblue",
             hjust = -0.5, vjust = 0.7)

## find the node shared by provided node labels
showNode(tree = tinyTree, only.leaf = TRUE,
         use.alias = FALSE)

showNode(tree = tinyTree, only.leaf = FALSE,
         use.alias = FALSE)
```

signalNode

Join nodes

Description

`joinNode` is to use as few as possible nodes to represent the provided nodes so that descendant leaves covered by the input nodes and output nodes are exactly the same.

Usage

```
signalNode(tree, node, use.alias = FALSE)

joinNode(tree, node, use.alias = FALSE)
```

Arguments

<code>tree</code>	A tree (phylo object)
<code>node</code>	A vector of node numbers or node labels

`use.alias` A logical value, TRUE or FALSE. The default is FALSE, and the node label would be used to name the output; otherwise, the alias of node label would be used to name the output. The alias of node label is created by adding a prefix "alias_" to the node number.

Value

A vector of nodes. The numeric value is the node number, and the vector name is the corresponding node label. If a node has no label, it would have NA as name when `use.alias = FALSE`, and have the alias of node label as name when `use.alias = TRUE`.

Author(s)

Ruizhu Huang

Examples

```
data(tinyTree)
library(ggtree)

# PLOT tree
# The node labels are in orange texts and the node numbers are in blue
ggtree(tinyTree, branch.length = 'none') +
  geom_text2(aes(label = label), color = "darkorange",
    hjust = -0.1, vjust = -0.7) +
  geom_text2(aes(label = node), color = "darkblue",
    hjust = -0.5, vjust = 0.7)

## find the node shared by provided node labels
joinNode(node = c('t4', 't9'), tree = tinyTree)
joinNode(node = c('t4', 't9'), tree = tinyTree)
joinNode(node = c('t10', 'Node_18', 't8'),
  tree = tinyTree,
  use.alias = FALSE)
joinNode(node = c('t10', 'Node_18', 't8'),
  tree = tinyTree,
  use.alias = TRUE)

## find the node shared by provided node numbers
joinNode(node = c(2, 3), tree = tinyTree)
joinNode(node = c(2, 3, 16), tree = tinyTree)
```

tinyTree

A simulated phylogenetic tree with 10 tips and 9 internal nodes

Description

A random phylo object created using the function [rtree](#)

Usage

```
tinyTree
```

Format

A phylo object with 10 tips and 9 internal nodes:

Tip labels t1, t2, ..., t10.

Node labels Node_11, Node_12, ..., Node_19

toTree

Translate a data frame to a phylo object

Description

toTree translates a data frame to a phylo object

Usage

```
toTree(data, column_order = NULL)
```

Arguments

data	A data frame or matrix.
column_order	A vector that includes the column names of data to reorder columns of data. Default is NULL, the original order of data is kept.

Details

The last column is used as the leaf nodes

Value

a phylo object

Author(s)

Ruizhu HUANG

Examples

```

library(ggtree)
# Example 1:
taxTab <- data.frame(R1 = rep("A", 5),
                     R2 = c("B1", rep("B2", 4)),
                     R3 = paste0("C", 1:5))
# Internal nodes: their labels are prefixed with colnames of taxTab
# e.g., R2:B2
tree <- toTree(data = taxTab)
# viz the tree
ggtree(tree) +
  geom_text2(aes(label = label), color = "red", vjust = 1) +
  geom_nodepoint()

# Example 2: duplicated rows in the 3rd and 4th rows
taxTab <- data.frame(R1 = rep("A", 5),
                     R2 = c("B1", rep("B2", 4)),
                     R3 = c("C1", "C2", "C3", "C3", "C4"))
# duplicated rows are removed with warnings
tree <- toTree(data = taxTab)

# Example 3: NA values in R2 column
# results: the internal node with the label 'R2:'
taxTab <- data.frame(R1 = rep("A", 5),
                     R2 = c("B1", rep("B2", 2), NA, "B2"),
                     R3 = c("C1", "C2", "C3", NA, "C4"))
tree <- toTree(data = taxTab)
# viz the tree
ggtree(tree) +
  geom_text2(aes(label = label), color = "red", vjust = 1) +
  geom_nodepoint()

# Example 4: duplicated values in the leaf column (R4)
# Not allowed and give errors
# taxTab <- data.frame(R1 = rep("A", 5),
#                      R2 = c("B1", rep("B2", 3), "B3"),
#                      R3 = c("C1", "C2", "C3", "C3", NA),
#                      R4 = c("D1", "D2", "D3", NA, NA))

# Example 5: loops caused by missing values in B2-C4, B3-C4
taxTab <- data.frame(R1 = rep("A", 6),
                     R2 = c("B1", rep("B2", 4), "B3"),
                     R3 = c("C1", "C2", "C3", "C3", "C4", "C4"),
                     R4 = c("D1", "D2", "D3", "D3", "D4", "D4"),
                     R5 = paste0("E", 1:6))
# resolve loops before run to Tree
# Suffix are adding to C4
taxNew <- resolveLoop(taxTab)
tree <- toTree(data = taxNew)

# viz the tree
ggtree(tree) +

```

```
geom_text2(aes(label = label), color = "red", vjust = 1) +  
geom_nodepoint()
```

trackNode	<i>track the nodes of a phylo tree</i>
-----------	--

Description

trackNode track nodes of a phylo tree by adding the alias labels to them

Usage

```
trackNode(tree)
```

Arguments

tree	A phylo object
------	----------------

Value

a phylo object

Author(s)

Ruizhu Huang

Examples

```
library(ggtree)  
  
data(tinyTree)  
  
ggtree(tinyTree, branch.length = 'none') +  
  geom_text2(aes(label = label), hjust = -0.3) +  
  geom_text2(aes(label = node), vjust = -0.8,  
             hjust = -0.3, color = 'blue')  
  
#check whether the node number and node label are matched  
trackTree <- trackNode(tinyTree)  
ggtree(trackTree, branch.length = 'none') +  
  geom_text2(aes(label = label), hjust = -0.3) +  
  geom_text2(aes(label = node), vjust = -0.8,  
             hjust = -0.3, color = 'blue')
```

transNode	<i>Transfer between node number and node label</i>
-----------	--

Description

convertNode does the transformation between the number and the label of a node on a tree

Usage

```
transNode(tree, node, use.alias = FALSE, message = FALSE)
```

```
convertNode(tree, node, use.alias = FALSE, message = FALSE)
```

Arguments

tree	A phylo object
node	A character or numeric vector representing tree node label(s) or tree node number(s)
use.alias	A logical value, TRUE or FALSE. This is an optional argument that only required when the input node is a numeric vector. The default is FALSE, and the node label would be returned; otherwise, the alias of node label would be output. The alias of node label is created by adding a prefix "alias_" to the node number.
message	A logical value, TRUE or FALSE. The default is FALSE. If TRUE, message will show when a tree have duplicated labels for some internal nodes.

Value

a vector

Author(s)

Ruizhu Huang

Examples

```
library(ggtree)

data(tinyTree)

ggtree(tinyTree, branch.length = 'none') +
  geom_text2(aes(label = label), hjust = -0.3) +
  geom_text2(aes(label = node), vjust = -0.8,
    hjust = -0.3, color = 'blue')

#check whether the node number and node label are matched
convertNode(tinyTree, node = c(11, 2, 4, 15))
```

```
convertNode(tree = tinyTree, node = c("Node_16", "Node_11"))
convertNode(tree = tinyTree, node = c("alias_16", "alias_11"))
```

TreeSummarizedExperiment-class

An S4 class TreeSummarizedExperiment

Description

The class **TreeSummarizedExperiment** is an extension class of standard [SingleCellExperiment](#) class. It has four more slots that are not in [SingleCellExperiment](#) class: rowTree, rowLinks, colTree and colLinks. The hierarchical information of rows (columns) is stored in rowTree (colTree) and the link between the rows (columns) of assays tables and nodes of the tree is given in rowLinks (colLinks).

Details

The class **TreeSummarizedExperiment** is designed to store rectangular data for entities (e.g., microbes or cell types) (assays), information about the hierarchical structure (rowTree on rows; colTree on columns), and the mapping information between the tree nodes and the rows or the columns of the rectangular data. Users could provide the hierarchical structure of the rows, columns or both) of the assays tables, and the link data will be automatically generated in rowLinks, colData or both, respectively. It's required that the object in rowLinks or colLinks has the LinkDataFrame class. Please see the page [LinkDataFrame](#) for more details.

Slots

rowTree A phylo object or NULL. It gives information about the hierarchical structure of rows of assays tables.

colTree A phylo object or NULL. It gives information about the hierarchical structure of columns of assays tables.

rowLinks A LinkDataFrame. It gives information about the link between the nodes of the rowTree and the rows of assays tables.

colLinks A LinkDataFrame. It gives information about the link between the nodes of the colTree and the columns of assays tables.

referenceSeq A DNASTringSet/DNASTringSetList object or some object coercible to a DNASTringSet/DNASTringSetList object. See [DNASTringSet](#) for more details.

... Other slots from [SingleCellExperiment](#)

Constructor

See [TreeSummarizedExperiment-constructor](#) for constructor functions.

Accessor

See [TreeSummarizedExperiment-accessor](#) for accessor functions.

See Also

[TreeSummarizedExperiment](#) [TreeSummarizedExperiment-accessor](#) [SingleCellExperiment](#)

TreeSummarizedExperiment-constructor

Construct a TreeSummarizedExperiment object

Description

TreeSummarizedExperiment constructs a TreeSummarizedExperiment object.

Usage

```
TreeSummarizedExperiment(
  ...,
  rowTree = NULL,
  colTree = NULL,
  rowNodeLab = NULL,
  colNodeLab = NULL,
  referenceSeq = NULL
)
```

Arguments

...	Arguments passed to the SummarizedExperiment constructor to fill the slots of the base class.
rowTree	A phylo object that provides hierarchical information of rows of assay tables.
colTree	A phylo object that provides hierarchical information of columns of assay tables.
rowNodeLab	A character string. It provides the labels of nodes that the rows of assays tables corresponding to. If NULL (default), the row names of the assays tables are used.
colNodeLab	A character string. It provides the labels of nodes that the columns of assays tables corresponding to. If NULL (default), the column names of the assays tables are used.
referenceSeq	A DNASTringSet/DNASTringSetList object or some object coercible to a DNASTringSet/DNASTringSetList object. See DNASTringSet for more details.

Details

The output TreeSummarizedExperiment object has very similar structure as the [SingleCellExperiment](#). The differences are summarized be as below.

- **rowTree** A slot exists in TreeSummarizedExperiment but not in SingleCellExperiment. It stores the tree structure(s) that provide(s) hierarchical information of assays rows or columns or both.

- **rowData** If a phylo object is available in the slot `treeData` to provide the hierarchical information about the rows of the assays table, the `rowData` would be a [LinkDataFrame-class](#) instead of [DataFrame](#). The data on the right side of the vertical line provides the link information between the assays rows and the tree phylo object, and could be accessed via `linkData`; The data on the left side is the original `rowData` like `SingleCellExperiment` object.
- **colData** Similar to the explanation for **rowData** as above.

More details about the `LinkDataFrame` in the `rowData` or `colData`.

- `nodeLab` The labels of nodes on the tree.
- `nodeLab\_alias` The alias of node labels on the tree.
- `nodeNum` The numbers of nodes on the tree.
- `isLeaf` It indicates whether the node is a leaf node or internal node.

Value

a `TreeSummarizedExperiment` object

Author(s)

Ruizhu HUANG

See Also

[TreeSummarizedExperiment](#) [TreeSummarizedExperiment-accessor](#) [SingleCellExperiment](#)

Examples

```
data("tinyTree")

# the count table
count <- matrix(rpois(100, 50), nrow = 10)
rownames(count) <- c(tinyTree$tip.label)
colnames(count) <- paste("C_", 1:10, sep = "_")

# The sample information
sampC <- data.frame(condition = rep(c("control", "trt"), each = 5),
                    gender = sample(x = 1:2, size = 10, replace = TRUE))
rownames(sampC) <- colnames(count)

# build a TreeSummarizedExperiment object
tse <- TreeSummarizedExperiment(assays = list(count),
                               colData = sampC,
                               rowTree = tinyTree)
```

TreeSummarizedExperiment-internal

TreeSummarizedExperiment internals

Description

Internal functions which should only be used in TreeSummarizedExperiment

Usage

```
## S4 method for signature 'TreeSummarizedExperiment'
vertical_slot_names(x)
```

unionLeaf

list leaf nodes that are the descendants of at least one specified node

Description

unionLeaf list the leaf nodes that are the descendants of (at least one) specified nodes.

Usage

```
unionLeaf(tree, node)
```

Arguments

tree	A phylo object.
node	A numeric or character vector. It specifies internal nodes that are changed to leaves via their node labels or numbers.

Value

A phylo object.

Examples

```
library(ggtree)
data(tinyTree)
ggtree(tinyTree, ladderize = FALSE) +
  geom_text2(aes(label = label), color = "darkorange",
    hjust = -0.1, vjust = -0.7) +
  geom_text2(aes(label = node), color = "darkblue",
    hjust = -0.5, vjust = 0.7) +
  geom_highlight(node = 18) +
  geom_point2()
```

```
u1 <- unionLeaf(tree = tinyTree, node = c(19, 17))
u2 <- unionLeaf(tree = tinyTree, node = c(19, 17, 7))
(u3 <- unionLeaf(tree = tinyTree, node = c(11, 17, 7)))
```

updateObject, TreeSummarizedExperiment-method

Update a TreeSummarizedExperiment object

Description

Update TreeSummarizedExperiment objects to the latest version of the class structure. This is usually called by methods in the TreeSummarizedExperiment package rather than by users or downstream packages.

Usage

```
## S4 method for signature 'TreeSummarizedExperiment'
updateObject(object, ..., verbose = FALSE)
```

Arguments

object	A TreeSummarizedExperiment object
...	additional arguments, for use in specific updateObject methods.
verbose	TRUE or FALSE, indicating whether information about the update should be reported.

Value

An updated TreeSummarizedExperiment object

Index

* datasets

tinyTree, [42](#)

* internal

.all_equal_in_list, [4](#)

.all_have_DNAStringSet, [4](#)

.all_have_DNAStringSetList, [4](#)

.all_nonnull_in_list, [5](#)

.all_null_in_list, [5](#)

.any_null_in_list, [5](#)

.auto_rename_list, [5](#)

.bind_link_tree, [6](#)

.is_equal_link, [6](#)

.match_phylo, [6](#)

.match_phylo_list, [6](#)

.match_x_dupY, [7](#)

.name_y_with_x, [7](#)

.numeric_ij, [8](#)

.rbind_refSeq, [8](#)

.replace_link_tree_1d, [9](#)

.subset_leaf, [9](#)

.update_whichTree, [10](#)

phylo, [31](#)

TreeSummarizedExperiment-internal,
[50](#)

.all_equal_in_list, [4](#)

.all_have_DNAStringSet, [4](#)

.all_have_DNAStringSetList, [4](#)

.all_nonnull_in_list, [5](#)

.all_null_in_list, [5](#)

.any_null_in_list, [5](#)

.auto_rename_list, [5](#)

.bind_link_tree, [6](#)

.is_equal_link, [6](#)

.match_phylo, [6](#)

.match_phylo_list, [6](#)

.match_x_dupY, [7](#)

.name_y_with_x, [7](#)

.numeric_ij, [8](#)

.rbind_refSeq, [8](#)

.replace_link_tree_1d, [9](#)

.subset_leaf, [9](#)

.update_whichTree, [10](#)

[,TreeSummarizedExperiment,ANY,ANY,ANY-method
(rowLinks), [34](#)

[<-,TreeSummarizedExperiment,ANY,ANY,TreeSummarizedExperim
(rowLinks), [34](#)

addLabel, [10](#)

aggTSE, [11](#), [14](#)

aggValue, [13](#)

apply, [12](#), [14](#)

asLeaf, [15](#)

asPhylo, [16](#)

BiocGenerics, [37](#)

BiocParallelParam, [12](#)

cbind, [32](#)

cbind,TreeSummarizedExperiment-method
(rbind,TreeSummarizedExperiment-method),
[32](#)

changeTree, [17](#)

colLinks(rowLinks), [34](#)

colLinks,TreeSummarizedExperiment-method
(rowLinks), [34](#)

colnames<-,TreeSummarizedExperiment-method
(rowLinks), [34](#)

colTree(rowLinks), [34](#)

colTree,TreeSummarizedExperiment-method
(rowLinks), [34](#)

colTree<-(rowLinks), [34](#)

colTree<-,TreeSummarizedExperiment-method
(rowLinks), [34](#)

colTreeNames(rowLinks), [34](#)

colTreeNames,TreeSummarizedExperiment-method
(rowLinks), [34](#)

colTreeNames<-(rowLinks), [34](#)

colTreeNames<-,TreeSummarizedExperiment-method
(rowLinks), [34](#)

- convertNode (transNode), 46
- countLeaf, 19
- countNode, 20
- DataFrame, 28, 49
- detectLoop, 21
- distNode, 22
- DNAStringSet, 37, 47, 48
- findAncestor, 23
- findChild, 24
- findDescendant (findOS), 25
- findOS, 25
- findSibling, 26
- isLeaf, 27
- joinNode (signalNode), 41
- LinkDataFrame, 28, 47
- LinkDataFrame
 - (LinkDataFrame-constructor), 28
- LinkDataFrame-class, 28
- LinkDataFrame-constructor, 28
- makeTSE, 29
- matTree, 30
- phylo, 31
- printNode, 31
- rbind, TreeSummarizedExperiment-method, 32
- referenceSeq (rowLinks), 34
- referenceSeq, TreeSummarizedExperiment-method (rowLinks), 34
- referenceSeq<- (rowLinks), 34
- referenceSeq<-, TreeSummarizedExperiment-method (rowLinks), 34
- resolveLoop, 33
- rowLinks, 34
- rowLinks, TreeSummarizedExperiment-method (rowLinks), 34
- rownames<-, TreeSummarizedExperiment-method (rowLinks), 34
- rowTree (rowLinks), 34
- rowTree, TreeSummarizedExperiment-method (rowLinks), 34
- rowTree<- (rowLinks), 34
- rowTree<-, TreeSummarizedExperiment-method (rowLinks), 34
- rowTreeNames (rowLinks), 34
- rowTreeNames, TreeSummarizedExperiment-method (rowLinks), 34
- rowTreeNames<- (rowLinks), 34
- rowTreeNames<-, TreeSummarizedExperiment-method (rowLinks), 34
- rtree, 42
- shareNode, 39
- showNode, 40
- signalNode, 41
- SingleCellExperiment, 34, 37, 47–49
- subsetByLeaf (rowLinks), 34
- subsetByLeaf, TreeSummarizedExperiment-method (rowLinks), 34
- subsetByNode (rowLinks), 34
- subsetByNode, TreeSummarizedExperiment-method (rowLinks), 34
- SummarizedExperiment, 48
- tinyTree, 42
- toTree, 43
- trackNode, 45
- transNode, 46
- TreeSummarizedExperiment-package, 3
- TreeSummarizedExperiment, 3, 12, 37, 48, 49
- TreeSummarizedExperiment
 - (TreeSummarizedExperiment-constructor), 48
- TreeSummarizedExperiment-accessor (rowLinks), 34
- TreeSummarizedExperiment-class, 47
- TreeSummarizedExperiment-combine (rbind, TreeSummarizedExperiment-method), 32
- TreeSummarizedExperiment-constructor, 48
- TreeSummarizedExperiment-internal, 50
- unionLeaf, 50
- updateObject, TreeSummarizedExperiment-method, 51
- vertical_slot_names, TreeSummarizedExperiment-method (TreeSummarizedExperiment-internal), 50