Package 'rrnni'

May 9, 2026

Title Manipulate with RNNI Tree Space

Version 0.1.1

Date 2023-08-26

Description Calculate RNNI distance between and manipulate with ranked trees. RNNI stands for Ranked Nearest Neighbour Interchange and is an extension of the classical NNI space (space of trees created by the NNI moves) to ranked trees, where internal nodes are ordered according to their heights (usually assumed to be times). The RNNI distance takes the tree topology into account, as standard NNI does, but also penalizes changes in the order of internal nodes, i.e. changes in the order of times of evolutionary events. For more information about the RNNI space see: Gavryushkin et al. (2018) <doi:10.1007/s00285-017-1167-9>, Collienne & Gavryushkin (2021) <doi:10.1007/s00285-021-01567-5>, Collienne et al. (2021) <doi:10.1007/s00285-021-01685-0>, and Collienne (2021) <http://hdl.handle.net/10523/12606>.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.2.3

Imports methods, ape

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

URL https://rrnni.biods.org/

BugReports https://github.com/biods/rrnni/issues

NeedsCompilation yes

Author Jiří C. Moravec [aut, cre],
Lena Collienne [aut]

Maintainer Jiří C. Moravec <jiri.c.moravec@gmail.com>

Repository CRAN

Date/Publication 2023-08-25 13:50:02 UTC

Contents

`as_ranked` *Convert a tree to ranked tree*

Description

Convert a tree of class "phylo" to a ranked tree of class "rankedPhylo".

Usage

```
as_ranked(x)
```

Arguments

`x`	a tree of class "phylo"

Details

To convert a tree to a ranked tree, the tree needs to be binary (i.e., fully resolved), be rooted, and ultrametric. Typically, such tree would be produced by coalescent process.

Ranked trees are similar to time-trees, only instead of time, we care about the order of the splits.

Value

a ranked tree of class "rankedPhylo"

Examples

```
# this will throw an error, the tree is not ultrametric
x = ape::rtree(5)
try(as_ranked(x))

# coalescent trees can be converted without problem
y = ape::rcoal(5)
as_ranked(y)
```

common_tips *Find shared tip labels*

Description

Find shared tip labels of a collection of trees. This function is useful when pruning a collection of related trees to a trees of same size and taxa.

Usage

```
common_tips(x)
```

Arguments

x	a collection of trees of class "multiPhylo"

Value

intersection of tip labels across all trees

Examples

```
trees = rankedPhylo(3:7)
common_tips(trees)
```

keep_tips *Prune a tree*

Description

Prune a tree or a collection of trees and keep only requested tips.

Usage

```
keep_tips(x, tips)

## S3 method for class 'phylo'
keep_tips(x, tips)

## S3 method for class 'multiPhylo'
keep_tips(x, tips)

## S3 method for class 'rankedPhylo'
keep_tips(x, tips)
```

Arguments

x	a tree of class "phylo" or "multiPhylo"
tips	tip labels to keep

Value

a pruned tree or a collection

Examples

```
tree = rankedPhylo(5)
# select randomly 3 tips to keep
tips = sample(tips(tree), 3)
keep_tips(tree, tips)

trees = rankedPhylo(3:7)
# get tips from the first tree
tips = tips(trees[[1]])
# prune all trees, all of them will have 3 tips
keep_tips(trees, tips)
```

---

random_tree *Create a random ranked phylogeny*

Description

Create a random ranked phylogeny using the coalescent method.

Usage

```
random_tree(n)
```

Arguments

n	the number of tips

Details

This is implementation of a ranked coalescent algorithm described in Collienne (2021). Starting from n tips of a tree, all with rank 0, randomly select two tips and merge them into a new node with rank 1, and add the new node to remaining tips. In next iteration, assign rank 2 and so on, until only a single node, the root, remains.

Value

random ranked tree of class "rankedPhylo"

References

Collienne, L. (2021). Spaces of Phylogenetic Time Trees (p. 158). University of Otago.

Examples

```
random_tree(5)
```

`rankedPhylo` *Create a ranked tree*

Description

Create a new, or coerce character vector or an existing tree into a ranked tree.

Usage

```
rankedPhylo(x)

## Default S3 method:
rankedPhylo(x)

## S3 method for class 'numeric'
rankedPhylo(x)

## S3 method for class 'character'
rankedPhylo(x)

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

## S3 method for class 'multiPhylo'
rankedPhylo(x)
```

Arguments

`x`	a numeric vector, character vector or a object of class `phylo` or `multiPhylo`, see details.

Details

This is a wrapper for multiple functions that create a ranked tree. As such, it accepts multiple types of inputs and produce either `rankedPhylo` or a collated list of ranked trees as a `multiRankedPhylo`.

If `x` is a numeric vector, create `n = length(x)` random ranked tree using the `random_tree` function. If `x` is a character vector, try to read the character as a newick-formatted tree. If `x` is object of a class `phylo` or `multiPhylo`, coerce these objects into `rankedPhylo` or `multiRankedPhylo` using the `as_ranked` function.

Value

ranked tree or trees of class `rankedPhylo` or `multiRankedPhylo`

Examples

```
# Create a single random ranked tree with 5 tips
rankedPhylo(5)

# Create multiple random ranked trees with 5 tips
rankedPhylo(c(5,5,5))

# Convert a coalescent tree into a ranked tree
x = ape::rcoal(5) # random coalescent tree
rankedPhylo(x)
```

`read_newick` *Read a newick tree*

Description

Read a tree in a newick format and convert it to a rankedPhylo

Usage

```
read_newick(x)
```

Arguments

`x`	a single character string containing tree in a newick format

Value

object of class "rankedPhylo"

Examples

```
read_newick("((A:1,B:1):1,C:2);")

# Note, not all valid newick trees are valid ranked trees
try( read_newick("((A,B),C);") )
```

rnni *Calculate RNNI distance*

Description

Calculate Ranked Nearest Neighbour Interchange (RNNI) distance.

Usage

```
rnni(x, y, normalize = FALSE)
```

Arguments

x	a tree of class "rankedPhylo"
y	a tree of class "rankedPhylo"
normalize	normalize the distance to the maximum distance of diameter (n-1)(n-2)/2 where n is the number of tips/leaves.

Details

The RNNI distance is the shortest distance between two trees in the RNNI space, which is defined by rank and NNI moves. This space is defined only for ranked trees. Non-ranked ultrametric trees can be coerced to ranked trees with as_ranked or rankedPhylo functions.

The distance can be normalized to the maximum possible distance, the diameter of the tree-space. The diameter is defined as (n-1)(n-2)/2, where n is the number of tips/leaves.

The used algorithm implements the TREEPATH algorithm described in Collienne (2021).

Value

an integer RNNI distance between x and y

References

Collienne, L. (2021). Spaces of Phylogenetic Time Trees. University of Otago.

sort_tips *Sort tip labels*

Description

Sort tip labels in a tree or a collection of trees.

Usage

```
sort_tips(x)

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

## S3 method for class 'multiPhylo'
sort_tips(x)
```

Arguments

x	a tree of class "phylo" or a collection of trees of class "multiPhylo"

Details

The rnni function assumes tip labels in an unambiguous order. This is due to the internal implementation not having tip labels, tip/leave are instead identified by the index in the node matrix. The sort_tips orders the tip labels and change the node indices in the node matrix so that the tips for two different trees of the same taxa have the same node id.

Value

an input tree with sorted tips

Examples

```
# generate random trees
# use rcoal(5) instead of rankedPhylo(5)
# as ranked phylo always generate tip.labels
# in the same order.
t1 = ape::rcoal(5) |> rankedPhylo()
t2 = ape::rcoal(5) |> rankedPhylo()
t1s = sort_tips(t1)
t2s = sort_tips(t2)
all(tips(t1s) == tips(t2s))

# for collection of trees:
trees = c(t1, t2)
sort_tips(trees)
```

tips *Extract tips from tree*

Description

Extract tip labels from the object of class "phylo" or "multiPhylo"

Usage

```
tips(x, ...)

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

## S3 method for class 'multiPhylo'
tips(x, all = FALSE, ...)
```

Arguments

x	an object of class "phylo" or "multiPhylo"
...	arguments to be passed to methods
all	**optional** extract tips for all trees, only if x is "multiPhylo"

Details

This is convenience method, it's purpose is to easily retrieve tip labels of a tree or a collection of trees with the same tip labels. Due to this, when called on a collection of trees (an object of class "multiPhylo"), only the first tree of a collection is accessed. To obtain tip labels from all trees, specify the argument `all=TRUE`.

Value

a vector of tip labels, or list of vectors in case of "multiPhylo" with `all=TRUE`

Examples

```
tree = rankedPhylo(5)
tips(tree)

trees = rankedPhylo(3:7)
# only the first tree is accessed
tips(trees)

# use this to obtain all tip labels
tips(trees, all=TRUE)
```

Index