Package ‘treedata.table’

October 9, 2025

Title Manipulation of Matched Phylogenies and Data using 'data.table'

Version 0.1.1

URL https://ropensci.github.io/treedata.table/,
https://docs.ropensci.org/treedata.table/,
https://github.com/ropensci/treedata.table/

BugReports https://github.com/ropensci/treedata.table/issues

Description An implementation that combines trait data and a phylogenetic tree (or trees) into a single object of class 'treedata.table'. The resulting object can be easily manipulated to simultaneously change the trait- and tree-level sampling. Currently implemented functions allow users to use a 'data.table' syntax when performing operations on the trait dataset within the 'treedata.table' object. For more details see Roman-Palacios et al. (2021) <doi:10.7717/peerj.12450>.

License MIT + file LICENSE

Depends R (>= 2.10), ape

Imports lazyeval, geiger, utils, data.table

RoxygenNote 7.3.2

Suggests covr, knitr, rmarkdown, testthat, utf8

VignetteBuilder knitr

Encoding UTF-8

NeedsCompilation no

Author Josef Uyeda [aut] (ORCID: <https://orcid.org/0000-0003-4624-9680>),
Cristian Roman-Palacios [aut, cre] (ORCID:
<https://orcid.org/0000-0003-1696-4886>),
April Wright [aut] (ORCID: <https://orcid.org/0000-0003-4692-3225>),
Luke Harmon [ctb],
Hugo Gruson [rev],
Kari Norman [rev]

Maintainer Cristian Roman-Palacios <cromanpa@arizona.edu>

Repository CRAN

Date/Publication 2025-10-09 12:10:06 UTC

Contents

`anolis` *Anole data*

Description

Anole data for treedata.table functions. Many of the traits (e.g. awesomeness, hostility) in this dataset retrieved from treeplyr are based on random numbers.

Usage

```
data(anolis)
```

Format

An object of class `list` of length 2.

Author(s)

Luke Harmon

`as.treedata.table` *Combine tree (or set of trees) and data.frame into a single treedata.table object*

Description

This function takes as input a tree of class `phylo` or `multiPhylo` and a `data.frame` and combines them into a treedata.table. If a `multiPhylo` is provided, all trees must have the same tip.labels. `treedata.table` object is sorted such that the rows in the data.table are matched to the tip.labels of the phylogeny. Tip.labels on the tree must match a column of tip names in the input data.frame. The output of this function will be a treedata.table, which can be manipulated as a data.table.

Usage

```
as.treedata.table(tree, data, name_column = "detect")
```

Arguments

`tree`	A tree of class `phylo` or multiple trees of class `multiPhylo`
`data`	A dataset in format `data.frame`
`name_column`	A character indicating the name of taxa in `data.frame`. If set to `detect` (default) `as treedata.table` will auto-detect this column

Value

An object of type `treedata.table` containing the tree and data.table

Examples

```
data(anolis)
anolis2 <- anolis$phy
anolis2$tip.label[1] <- "NAA"
anolis1 <- anolis$phy
anolis1$tip.label[1] <- "NAA"
trees <- list(anolis1, anolis2)
class(trees) <- "multiPhylo"
treesFM <- list(anolis$phy, anolis$phy)
class(treesFM) <- "multiPhylo"

# A phylo object that fully matches the data
td <- as.treedata.table(tree = anolis$phy, data = anolis$dat)
# A multiphylo object that fully matches the data
td <- as.treedata.table(tree = treesFM, data = anolis$dat)
# A phylo object that partially matches the data
td <- as.treedata.table(tree = anolis1, data = anolis$dat)
# A multiphylo object that partially matches the data
td <- as.treedata.table(tree = trees, data = anolis$dat)
```

`detectAllCharacters` *Apply detectCharacterType over an entire matrix*

Description

This function detects whether each column in a matrix is a continuous (e.g., with values 2.45, 9.35, and so on) or a discrete character (e.g., with values blue, red, yellow).

Usage

```
detectAllCharacters(mat, cutoff = 0.1)
```

Arguments

`mat`	A matrix of data
`cutoff`	Cutoff value for deciding if numeric data might actually be discrete: if nlev is the number of levels and n the length of dat, then nlev / n should exceed cutoff, or the data will be classified as discrete

Value

Vector of either "discrete" or "continuous" for each variable in matrix

Examples

```
data(anolis)
detectAllCharacters(anolis$dat)
```

`detectCharacterType` *Function to detect whether a character is continuous or discrete*

Description

This function detects whether a given vector is a continuous (e.g., with values 2.45, 9.35, and so on) or a discrete (e.g., with values blue, red, yellow) character.

Usage

```
detectCharacterType(dat, cutoff = 0.1)
```

Arguments

`dat`	A vector of data
`cutoff`	Cutoff value for deciding if numeric data might actually be discrete: if nlev is the number of levels and n the length of dat, then nlev / n should exceed cutoff, or the data will be classified as discrete

Value

Either "discrete" or "continuous"

Examples

```
data(anolis)
detectCharacterType(anolis$dat[, 1])
```

`droptreedata.table` *Function dropping taxa from an object of class* `treedata.table`

Description

This function can be used to remove species from an object of class `treedata.table`. The resulting `treedata.table` will include fully matching `$dat` and `$phy` elements. The user should confirm the changes before they are processed.

Usage

```
droptreedata.table(tdObject, taxa)
```

Arguments

`tdObject`	An object of class `treedata.table`
`taxa`	A vector class `character` containing all taxa that needs to be dropped from the original `treedata.table` object

Value

An object of class `treedata.table` with matching `$dat` and `$phy` elements based on whether `taxa` were dropped or not.

Examples

```
data(anolis)
# With a multiphylo object in the treedata.table object
td <- as.treedata.table(anolis$phy, anolis$dat)
droptreedata.table(
  tdObject = td, taxa =
    c("chamaeleonides", "eugenegrahami")
)

# With a multiphylo object in the treedata.table object
treesFM <- list(anolis$phy, anolis$phy)
class(treesFM) <- "multiPhylo"
td <- as.treedata.table(treesFM, anolis$dat)
droptreedata.table(
  tdObject = td, taxa =
    c("chamaeleonides", "eugenegrahami")
)
```

`extractVector` *Returning a named vector from a treedata.table object*

Description

Returning a named vector from a treedata.table object

Usage

```
extractVector(tdObject, ...)
```

Arguments

`tdObject`	A treedata.table object
`...`	The name of the column or columns to select.

Value

A named vector or a list of named vectors

Examples

```
data(anolis)
td <- as.treedata.table(tree = anolis$phy, data = anolis$dat)
# extracts the named vector for SVL from the td object
extractVector(td, "SVL")
# extracts the named vector for SVL and ecomorph from the td object
extractVector(td, "SVL", "ecomorph")
```

`filterMatrix` *Filter a character matrix, returning either all continuous or all discrete characters*

Description

This function filters a character matrix based on continuous (e.g., with values 2.45, 9.35, and so on) or discrete characters (e.g., with values blue, red, yellow).

Usage

```
filterMatrix(mat, returnType = "discrete")
```

Arguments

`mat`	A character matrix of class data.frame
`returnType`	Either discrete or continuous

Value

data.frame with only discrete (default) or continuous characters

Examples

```
data(anolis)
filterMatrix(anolis$dat, "discrete")
```

`forceNames` *Force names for rows, columns, or both*

Description

This function creates column names (`colnames`), row.names (`row.names`), or both in an unnamed `data.frame` or `matrix`.

Usage

```
forceNames(dat, nameType = "row")
```

Arguments

`dat`	A vector of data
`nameType`	either: **"row"** Rows (default) **"col"** Columns **"rowcol"** Both rows and columns

Value

An object of type ‘data.frame with labeled columns, rows, or both.

Examples

```
data(anolis)
forceNames(anolis$dat, "row")
```

`hasNames` *Row and column name check*

Description

This function checks whether a given `data.frame` or `matrix` has column names (`colnames`), row.names (`row.names`), or both.

Usage

```
hasNames(dat, nameType = "row")
```

Arguments

`dat`	A vector of data
`nameType`	either: **"row"** Rows (default) **"col"** Columns **"rowcol"** Both rows and columns

Value

`TRUE` or `FALSE` indicating if the object has names (`columns`, `rows`, or `both`)

Examples

```
data(anolis)
hasNames(anolis$dat, "row")
```

`head.treedata.table` *Return the first part of an treedata.table object*

Description

Return the first part of an treedata.table object

Usage

```
## S3 method for class 'treedata.table'
head(x, ...)
```

Arguments

`x`	a treedata.table object
`...`	Additional arguments passed to head.data.table

Value

First part of an treedata.table object

Examples

```
data(anolis)
td <- as.treedata.table(anolis$phy, anolis$dat)
head(td)
```

`print.treedata.table` *Print method treedata.table objects*

Description

Print method treedata.table objects

Usage

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

Arguments

`x`	an object of class "treedata.table"
`...`	additional arguments passed to "head.treedata.table"

Value

Function uses prints the tree and the first lines of the data.table object.

`pulltreedata.table` *Returns the character matrix or phylogeny from a treedata.table object*

Description

Returns the character matrix or phylogeny from a treedata.table object

Usage

```
pulltreedata.table(tdObject, type = c("dat", "phy"))
```

Arguments

`tdObject`	A treedata.table object
`type`	Whether the output of the function is a tree ('type="phylo"') or a data.table ('type="dat"')

Value

A `data.table` or `phylo` object from the original `treedata.table` object

Examples

```
data(anolis)
td <- as.treedata.table(anolis$phy, anolis$dat)
pulltreedata.table(td, type = "phy")
pulltreedata.table(td, type = "dat")
```

`summary.treedata.table` *Summarizing treedata.table objects*

Description

Summarizing treedata.table objects

Usage

```
## S3 method for class 'treedata.table'
summary(object, ...)
```

Arguments

`object`	an object of class "treedata.table"
`...`	additional arguments passed to "head.treedata.table"

Value

Function tries to be smart about summarizing the data and detecting continuous vs. discrete data, as well as whether any data have missing data. Also returns the taxa that are dropped from either the original tree or the original data.

Examples

```
data(anolis)
td <- as.treedata.table(anolis$phy, anolis$dat)
summary(td)
```

`tail.treedata.table` *Return the last part of an treedata.table object*

Description

Return the last part of an treedata.table object

Usage

```
## S3 method for class 'treedata.table'
tail(x, ...)
```

Arguments

`x`	a treedata.table object
`...`	Additional arguments passed to head.data.table

Value

Last part of an treedata.table object

Examples

```
data(anolis)
td <- as.treedata.table(anolis$phy, anolis$dat)
tail(td)
```

`tdt` *Run a function on a* `treedata.table` *object*

Description

Run a function on a `treedata.table` object

Usage

```
tdt(tdObject, ...)
```

Arguments

`tdObject`	A treedata.table object
`...`	A function call.

Details

This function allows R functions that use trees and data to be run on`treedata.table` objects.

Value

Function output for a single tree (phylo) or a list of function outputs (one per each tree in the MultiPhylo object)

Examples

```
data(anolis)


# A treedata.table object with a phylo $phy
td <- as.treedata.table(anolis$phy, anolis$dat)
tdt(td, geiger::fitContinuous(phy, extractVector(td, "SVL"),
  model = "BM", ncores = 1
))


# A treedata.table object with a multiPhylo $phy
treesFM <- list(anolis$phy, anolis$phy)
class(treesFM) <- "multiPhylo"
td <- as.treedata.table(treesFM, anolis$dat)
tdt(td, geiger::fitContinuous(phy, extractVector(td, "SVL"),
  model = "BM",
  ncores = 1
))
```

`[.treedata.table` *Function for performing data.table operations on an object of class* `treedata.table`

Description

This function can be used to subset rows, select and compute on columns data.table.

Usage

```
## S3 method for class 'treedata.table'
x[...]
```

Arguments

`x`	An object of class `treedata.table`
`...`	Arguments in the structure of `data.table` used to perform changes on the `treedata.table` object

Value

A new object of class `treedata.table` with `$dat` and `$phy` corresponding with the changes set to `$dat` using data.table's structure.

Examples

```
data(anolis)
anolis2 <- anolis$phy
anolis2$tip.label[1] <- "NAA"
anolis1 <- anolis$phy
anolis1$tip.label[1] <- "NAA"
trees <- list(anolis1, anolis2)
class(trees) <- "multiPhylo"
treesFM <- list(anolis$phy, anolis$phy)
class(treesFM) <- "multiPhylo"

# A phylo object that fully matches the data
td <- as.treedata.table(tree = anolis$phy, data = anolis$dat)
td <- as.treedata.table(anolis$phy, anolis$dat)
td[, SVL]
td[island == "Cuba" & ecomorph == "TG", .(ecomorph, island, SVL)]
td[, utils::head(.SD, 1), by = .(ecomorph, island)]

# A multiphylo object that fully matches the data
td <- as.treedata.table(tree = treesFM, data = anolis$dat)
td <- as.treedata.table(treesFM, anolis$dat)
td[, SVL]
td[island == "Cuba" & ecomorph == "TG", .(ecomorph, island, SVL)]
td[, utils::head(.SD, 1), by = .(ecomorph, island)]

# A phylo object that partially matches the data
td <- as.treedata.table(tree = anolis1, data = anolis$dat)
td <- as.treedata.table(anolis1, anolis$dat)
td[, SVL]
td[island == "Cuba" & ecomorph == "TG", .(ecomorph, island, SVL)]
td[, utils::head(.SD, 1), by = .(ecomorph, island)]

# A multiphylo object that partially matches the data
td <- as.treedata.table(tree = trees, data = anolis$dat)
td <- as.treedata.table(trees, anolis$dat)
td[, SVL]
td[island == "Cuba" & ecomorph == "TG", .(ecomorph, island, SVL)]
td[, utils::head(.SD, 1), by = .(ecomorph, island)]
```

`[[.treedata.table` *Function for extract a named vector from an object of class* `treedata.table`

Description

This function extracts a named vector for any trait from an object of class `treedata.table`.

Usage

```
## S3 method for class 'treedata.table'
x[[..., exact = TRUE]]
```

Arguments

`x`	An object of class `treedata.table`
`...`	Column name in class `character`
`exact`	whether exact search should be conducted

Value

A new object of class `vector` with names set to labels corresponding to tip labels in the provided `treedata.table` object.

Examples

```
data(anolis)
# With a phylo object
td <- as.treedata.table(anolis$phy, anolis$dat)
td[["SVL"]]

# With a multiPhylo object
treesFM <- list(anolis$phy, anolis$phy)
class(treesFM) <- "multiPhylo"
td <- as.treedata.table(treesFM, anolis$dat)
td[["SVL"]]
```

Index