Package 'treeplyr'

September 17, 2020

Type Package

Title 'dplyr' Functionality for Matched Tree and Data Objects

Version 0.1.10

Date 2020-09-04

Maintainer Josef Uyeda <juyeda@vt.edu>

Description Matches phylogenetic trees and trait data, and
allows simultaneous manipulation of the tree and data using 'dplyr'.

License GPL-2 | GPL-3

Depends ape (>= 3.0-6), dplyr (>= 0.8.0), R (>= 3.1.2)

Imports Rcpp (>= 0.10.3), lazyeval, phytools, geiger

LinkingTo Rcpp

URL https://github.com/uyedaj/treeplyr

BugReports https://github.com/uyedaj/treeplyr/issues

RoxygenNote 7.1.1

Encoding UTF-8

NeedsCompilation yes

Author Josef Uyeda [cre] (<https://orcid.org/0000-0003-4624-9680>),
Luke Harmon [aut] (<https://orcid.org/0000-0002-4985-5750>)

Repository CRAN

Date/Publication 2020-09-17 10:10:02 UTC

R topics documented:

`anolis` *Anole data*

Description

Anole data for aRbor functions

Usage

```
data(anolis)
```

Format

An object of class `list` of length 2.

`detectAllCharacters` *Apply detectCharacterType over an entire matrix*

Description

Apply detectCharacterType over an entire matrix

Usage

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

Arguments

`mat`	A matrix of data
`repeatsAsDiscrete`	If TRUE, consider numeric variables that repeat values exactly as discrete; see cutoff
`cutoff`	Cutoff value for deciding if numeric data might actually be descrete: 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

Function to detect whether a character is continuous or discrete

Usage

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

Arguments

`dat`	A vector of data
`repeatsAsDiscrete`	If TRUE, consider numeric variables that repeat values exactly as discrete; see cutoff
`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])
```

filter.treedata — *Function for filtering rows from an object of class* `treedata`

Description

This function can be used to select a subset of species (rows) from a treedata object; see `filter`.

Usage

```
## S3 method for class 'treedata'
filter(.data, ...)

## S3 method for class 'grouped_treedata'
filter(.data, ...)
```

Arguments

`.data`	An object of class `treedata`
`...`	Additional arguments to filter by

Value

An object of class `treedata` with the dataset filtered by the specified criteria.

See Also

`filter`

Examples

```
data(anolis)
td <- make.treedata(anolis$phy, anolis$dat, name_column=1)
tdfilter <- filter(td, island=="Cuba", SVL > 3.5)
```

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

Description

Filter a matrix, returning either all continuous or all discrete characters

Usage

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

Arguments

`mat`	A matrix of data
`charType`	A vector of character types (perhaps from detectAllCharacters)
`returnType`	Either discrete or continuous

Value

Matrix with only discrete or continuous characters

Examples

```
data(anolis)
aType<-detectAllCharacters(anolis$dat)
filterMatrix(anolis$dat, aType, "discrete")
```

`forceFactor` *Function for checking whether a treedata object contains only factors and for forcing data columns into factor format*

Description

This function can be used to check if a treedata object contains factors and, if desired, convert all columns automatically to factors.

Usage

```
forceFactor(tdObject, return.factor = TRUE)
```

Arguments

`tdObject`	A `treedata` object
`return.factor`	If TRUE, then a treedata object with all factors will be returned; columns will be forced into factors using `factor` and any with no repeated elements will be removed.

Value

If return.factor, then an object of class "`treedata`" with all columns as factors.

Examples

```
data(anolis)
td <- make.treedata(anolis$phy, anolis$dat)
tdforcefactor <- forceFactor(td)
```

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

Description

Force names for rows, columns, or both

Usage

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

Arguments

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

Examples

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

forceNumeric *Function for checking whether a treedata object contains only numeric columns and for forcing data columns into numeric format*

Description

This function can be used to check if a treedata object contains numeric columns and, if desired, drop all non-numeric columns.

Usage

```
forceNumeric(tdObject, return.numeric = TRUE)
```

Arguments

`tdObject`	A `treedata` object
`return.numeric`	If TRUE, then a treedata object with all numeric columns will be returned; non-numeric columns will be removed.

Value

If return.numeric, then an object of class "`treedata`" with only numeric columns.

Examples

```
data(anolis)
td <- make.treedata(anolis$phy, anolis$dat)
tdnumeric <- forceNumeric(td)
```

`getVector` *A function for returning a named vector from a data frame or matrix with row names*

Description

A function for returning a named vector from a data frame or matrix with row names

Usage

```
getVector(td, ...)
```

Arguments

`td`	A treedata object
`...`	The name of the column to select

Value

A named vector

`group_by.treedata` *Function for grouping an object of class* `treedata`

Description

This function can be used to group a treedata object by some factor.

Usage

```
## S3 method for class 'treedata'
group_by(.data, ..., .add = FALSE)

## S3 method for class 'grouped_treedata'
ungroup(x, ...)
```

Arguments

`.data`	An object of class `treedata`
`...`	The name of the grouping factor.
`.add`	By default, when .add = FALSE, group_by will override existing groups. To instead add to the existing groups, use .add = TRUE
`x`	An object of class `treedata`

Details

Groups the data frame and phylogeny by one of the factors in the data table.

Value

An object of class `grouped_treedata`.

See Also

`summarize`

Examples

```
data(anolis)
td <- make.treedata(anolis$phy, anolis$dat)
tdGrouped <- group_by(td, ecomorph)
summarize(tdGrouped, ntips = length(phy$tip.label),
   totalBL = sum(phy$edge.length), meanSVL = mean(SVL), sdSVL = sd(SVL))
```

`hasNames` *Row and column name check*

Description

Row and column name check

Usage

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

Arguments

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

Examples

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

`make.treedata` *Function for making an object of class* `treedata`

Description

This function generates an object of class `treedata` that ensures that the ordering of tip labels and data remain intact. The object can be manipulated using `dplyr` functions.

Usage

```
make.treedata(tree, data, name_column = "detect", as.is = FALSE)
```

Arguments

`tree`	An object of class 'phylo'
`data`	A data frame or matrix
`name_column`	An optional argument that specifies the column of `data` that contains the names to be matched to the tree. By default, it is set to "detect" which finds the column with the most matches to the tree (including the rownames).
`as.is`	Whether convert to factors. When FALSE (default), convert character vectors to factors.

Value

An object of class `"treedata"`. The tree is pruned of tips not represented in the data, and the data is filtered for taxa not in the tree. The data is returned as a data frame tble that is compatible with `dplyr` functions.

Examples

```
data(anolis)
td <- make.treedata(anolis$phy, anolis$dat)
```

mutate.treedata   *Function for mutating an object of class* `treedata`

Description

This function can be used to add new variables to a treedata object; see `mutate`.

Usage

```
## S3 method for class 'treedata'
mutate(.data, ...)

## S3 method for class 'grouped_treedata'
mutate(.data, ...)
```

Arguments

`.data`	An object of class `treedata`
`...`	Arguments to mutate the treedata object

Value

An object of class `treedata` with new data added.

See Also

`mutate`

Examples

```
data(anolis)
td <- make.treedata(anolis$phy, anolis$dat)
tdmutate <- mutate(td, lnSVL = log(SVL), badassery = awesomeness + hostility)
```

paint_clades   *Add regimes to a treedata object*

Description

This function paints clades on the phylogeny and adds a data column that specifies to which clade each species belongs

Usage

```
paint_clades(
  tdObject,
  nclades = 1,
  name = "clades",
  interactive = TRUE,
  type = "nodes",
  ids = NULL,
  plot = TRUE
)
```

Arguments

`tdObject`	A `treedata` object
`nclades`	The number of clades that will be specified if used interactively
`name`	The name of the resulting data column
`interactive`	If `TRUE`, then a plot will appear that will allow the user to click on `nclades` branches. The selections will then be coverted into the data table.
`type`	Either "nodes" or "branches" specifying if the ids provided specify the branch id (assuming a post-ordered tree) or the node number. Ignored if `interactive = TRUE`.
`ids`	A vector of node numbers of branch numbers that specify clades. Ignored if `interactive=TRUE`.
`plot`	If `TRUE` and `interactive = FALSE` then a simmap plot is produced.

Examples

```
data(anolis)
td <- make.treedata(anolis$phy, anolis$dat)
td <- reorder(td, "postorder")
td.painted <- paint_clades(td, interactive=FALSE, type="nodes",
                                    ids=c(184, 160, 135, 122), plot=TRUE)
td.painted <- group_by(td.painted, clades)
summarise(td.painted,
             psig1 = phytools::phylosig(setNames(SVL, phy$tip.label), tree=phy),
                   meanSVL = mean(SVL))
```

`reorder` *Reorder a* `treedata` *object*

Description

Reorders a `treedata` object. Both the tips and the data are automatically reordered to match.

Usage

```
reorder(tdObject, ...)

## S3 method for class 'treedata'
reorder(tdObject, order = "postorder", index.only = FALSE, ...)
```

Arguments

`tdObject`	An object of class `treedata`
`...`	Additional arguments to reorder.phylo
`order`	Method for reordering
`index.only`	Whether a index is returned rather than the reordered treedata object

Value

An object of class `treedata`

See Also

`reorder.phylo`

Examples

```
data(anolis)
td <- make.treedata(anolis$phy, anolis$dat)
td <- reorder(td, "postorder")
```

`select.treedata` *Function for selecting columns from an object of class* `treedata`

Description

This function can be used to select a subset of variables (columns) from a treedata object; see `select`.

Usage

```
## S3 method for class 'treedata'
select(.data, ...)
```

Arguments

`.data`	An object of class `treedata`
`...`	Additional arguments to select columns

Value

An object of class `treedata` with specified variables selected.

See Also

`select`

Examples

```
data(anolis)
td <- make.treedata(anolis$phy, anolis$dat)
tdselect <- select(td, SVL, awesomeness)
```

`slice.treedata` *Choose rows by their ordinal position in the tbl for an object of class* `treedata`

Description

This function can be used to drop tips from tree and data; see `slice`.

Usage

```
## S3 method for class 'treedata'
slice(.data, ...)
```

Arguments

`.data`	An object of class `treedata`
`...`	Integer row values

Value

An object of class `treedata`.

See Also

`slice`

Examples

```
data(anolis)
td <- make.treedata(anolis$phy, anolis$dat)
tdslice <- slice(td, 1:5)
tdslice
```

`summarise.treedata` *Function for summarizing an object of class* `treedata`

Description

This function can be used to summarize a treedata object.

Usage

```
## S3 method for class 'treedata'
summarise(.data, ...)

## S3 method for class 'grouped_treedata'
summarise(.data, ...)
```

Arguments

`.data`	An object of class `treedata`
`...`	Additional expressions by which to summarize data in the `treedata` object

Details

Summarizing `treedata` objects allows expressions using the objects phy. The `treedata` object can also be grouped, with summary statistics being applied to the pruned groups and phylogenies.

Value

An object of class `tbl_df` with the requested summary data.

See Also

`summarize`, `group_by`

Examples

```
data(anolis)
td <- make.treedata(anolis$phy, anolis$dat)
summarize(td, ntips = length(phy$tip.label), meanSVL = mean(SVL), sdSVL = sd(SVL))
tdGrouped <- group_by(td, ecomorph)
summarize(tdGrouped, ntips = length(phy$tip.label),
   totalBL = sum(phy$edge.length), meanSVL = mean(SVL), sdSVL = sd(SVL))
```

tdapply — *Apply a function over all treedata object columns and return a list of results, analogously to the normal apply function*

Description

Apply a function over all treedata object columns and return a list of results, analogously to the normal apply function

Usage

```
tdapply(tdObject, MARGIN, FUN, ...)
```

Arguments

`tdObject`	A treedata object
`MARGIN`	the margin over which the data is applied (e.g. 1 = rows, 2 = columns)
`FUN`	A function to apply over the data frame
`...`	Additional parameters passed on to FUN

Details

Note that if the parameter `phy` is specified in the additional parameters (i.e. '...'), then it will be substituted with the `treedata` object `$phy`.

Examples

```
data(anolis)
td <- make.treedata(anolis$phy, anolis$dat)
td %>% forceNumeric(.) %>% tdapply(., 2, phytools::phylosig, tree=phy)
```

treedply — *Run a function on a `treedata` object*

Description

Run a function on a `treedata` object

Usage

```
treedply(tdObject, ...)

## S3 method for class 'treedata'
treedply(tdObject, ...)
```

Arguments

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

Details

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

Value

Function output

Examples

```
data(anolis)
td <- make.treedata(anolis$phy, anolis$dat)
treedply(td, geiger::fitContinuous(phy, getVector(td, SVL), model="BM", ncores=1))
treedply(td, phytools::phylosig(phy, getVector(td, awesomeness), "lambda", test=TRUE))
treedply(td, phytools::phenogram(phy, getVector(td, SVL), ftype="off", spread.labels=FALSE))
```

`treeply` *Run a function on the phylogeny of a* `treedata` *object*

Description

Applies a function to the phylogeny in a `treedata` object. If the order of tips are changed, or if tips are dropped, then the data are automatically reordered to match the tree.

Usage

```
treeply(tdObject, ...)

## S3 method for class 'treedata'
treeply(tdObject, FUN, ...)
```

Arguments

`tdObject`	An object of class `treedata`
`...`	Additional arguments
`FUN`	A function that operates on an object of class 'phylo'

Value

An object of class `treedata`

Examples

```
data(anolis)
td <- make.treedata(anolis$phy, anolis$dat)
td2 <- treeply(td, drop.tip, 1:50)

par(mfrow=c(1,2))
plot(td$phy)
plot(td2$phy)
```

`treeplyr` *treeplyr: 'dplyr' Functionality for Matched Tree and Data Objects*

Description

Matches phylogenetic trees and trait data, and allows simultaneous manipulation of the tree and data using 'dplyr'.

Author(s)

Josef Uyeda

`treeplyr-defunct` *Defunct functions in treeplyr*

Description

These functions have been removed to reflect changes in dplyr.

Details

- `group_by_.treedata`: This function is defunct, use group_by instead.
- `mutate_.treedata`, `mutate_.grouped_treedata`: This function is defunct, use mutate instead.
- `slice_.treedata`: This function is defunct, use slice instead.
- `select_.treedata`: This function is defunct, use select instead.
- `filter_.treedata`,`filter_.grouped_treedata`: This function is defunct, use filter instead.

Index