Package ‘turner’

April 4, 2024

Title Turn Vectors and Lists of Vectors into Indexed Structures

Version 0.1.9

Date 2024-04-04

Author Frederic Bertrand [cre] (<https://orcid.org/0000-0002-0837-8281>), Gaston Sanchez [aut]

Maintainer Frederic Bertrand <frederic.bertrand@utt.fr>

Description Package designed for working with vectors and lists of vectors, mainly for turning them into other indexed data structures.

Encoding UTF-8

LazyLoad yes

NeedsCompilation no

RoxygenNote 7.2.3

URL https://fbertran.github.io/turner/, https://github.com/fbertran/turner/

BugReports https://github.com/fbertran/turner/issues/

Depends R (>= 3.0)

Imports tester

Suggests testthat, knitr

VignetteBuilder knitr

License GPL-3

Collate 'df_to_blocks.r' 'dummy_to_list.r' 'factor_to_dummy.r' 'from_to.r' 'funlist.r' 'indexify.r' 'lengths.r' 'list_ones.r' 'list_to_dummy.r' 'list_to_matrix.r' 'listify.r' 'listsize.r' 'matrix_to_blocks.r' 'vector_to_dummy.r' 'turner.r'

Repository CRAN

Date/Publication 2024-04-04 08:00:02 UTC

R topics documented:

`df_to_blocks` *Split a data frame into blocks*

Description

Split a data frame into a list of blocks (either by rows or by columns)

Usage

```
df_to_blocks(DataFrame, blocks, byrow = TRUE)
```

Arguments

`DataFrame`	a data frame to split
`blocks`	either a list or a vector indicating the blocks. If `blocks` is a list of vectors, then the length of each vector defines the size of the blocks. If `blocks` is a vector, then each element represents the size of the blocks.
`byrow`	logical. If `TRUE` (the default) the data frame is split by rows, otherwise the data frame is split by columns

Value

A list of data frames

Author(s)

Gaston Sanchez

See Also

`matrix_to_blocks`

Examples

```
# say you have a data frame
iris_df = iris[c(1:3,51:53,101:103),]

# list defining the blocks
row_blocks = list(1:3, 4:6, 7:9)
col_blocks = c(2, 2, 1)

# split data into list of blocks (by rows)
df_to_blocks(iris_df, row_blocks)

# split data into list of blocks (by columns)
df_to_blocks(iris_df, col_blocks, byrow=FALSE)
```

`dummy_to_list` *Create an indexed list from a dummy matrix*

Description

Create an indexed list from the columns of a dummy (or semi-dummy) matrix

Usage

```
dummy_to_list(Dummy)
```

Arguments

`Dummy`	matrix (dummy by columns)

Value

A list of indexed vectors

Author(s)

Gaston Sanchez

See Also

`list_to_dummy`, `listify`

Examples

```
# let's say you have a list like this
some_list = list(1:3, 1:2, 1:4)

# first create a dummy matrix based on some_list
some_dummy = list_to_dummy(some_list)

# now apply 'dummy_to_list'
dummy_to_list(some_dummy)

# a semi-dummy matrix
semi_dummy = some_dummy
semi_dummy[semi_dummy != 0] = rnorm(listsize(some_list))
dummy_to_list(semi_dummy)
```

`factor_to_dummy` *Create a dummy matrix from the elements in a factor*

Description

Create a dummy matrix based on the elements of a factor. Each column in the produced matrix is a dummy indicator.

Usage

```
factor_to_dummy(afactor)
```

Arguments

`afactor`	a factor (preferably of vectors)

Value

A matrix of dummy variables

Author(s)

Gaston Sanchez

See Also

`vector_to_dummy`, `list_to_dummy`

Examples

```
# let's say you have a list like this
some_factor = iris$Species[c(1:3,51:53,101:103)]

# get dummy matrix
factor_to_dummy(some_factor)
```

`from_to` *Starting and ending positions*

Description

Get the starting position 'from' and the ending position 'to' of the elements contained in a vector (or a list of vectors)

Usage

```
from_to(x, ...)
```

Arguments

`x`	a numeric vector or a list of vectors
`...`	further arguments are ignored

Value

A list with two vectors: '$from' and '$to'. '$from' contains the indices with starting positions. '$to' contains the indices with ending positions.

Author(s)

Gaston Sanchez

See Also

`lengths`, `listsize`

Examples

```
# let's say you have a numeric vector like this
num_vec = c(2, 3, 1, 4)

# get 'from' and 'to' indices
start_end = from_to(num_vec)
from = start_end$from
to = start_end$to

#' let's say you have a list like this
str_list = list(c("a","b","c"), c("d", "e"), c("f","g","h"))

# get 'from' and 'to' indices
start_end = from_to(str_list)
from = start_end$from
to = start_end$to
```

`funlist` *Apply a function to all elements in a list*

Description

Applies a function to the unlisted elements of a list

Usage

```
funlist(alist, f, ...)
```

Arguments

`alist`	a list
`f`	a function to be applied
`...`	further arguments passed on to `f`

Value

value

Author(s)

Gaston Sanchez

See Also

`lapply`, `sapply`

Examples

```
# say you have some list
list1 = list(1:5, runif(3), rnorm(4))

# get the sum of all elements in list1
funlist(list1, sum)

# get the maximum element in list1
funlist(list1, max)

# say you have missing data
list2 = list(c(1:4, NA), runif(3), rnorm(4))

# get the sum removing NAs
funlist(list2, sum, na.rm=TRUE)
```

`indexify` *Create indices for elements in a vector or list*

Description

Create indexed components for the elements of a list.

Usage

```
indexify(x, out)
```

Arguments

`x`	a numeric vector or list of vectors
`out`	string indicating the output format ("`vector`" or "`list`")

Value

A vector (or list) of indexed numbers

Author(s)

Gaston Sanchez

See Also

`listify`

Examples

```
# let's say you have a numeric vector like this
num_vec = c(2, 3, 1, 4)

# get indices in vector format
indexify(num_vec)

# let's say you have a list like this
str_list = list(c("a","b","c"), c("d", "e"), c("f","g","h"))

# get indices in vector format
indexify(str_list)

# get indices in list format
indexify(str_list, "list")
```

`lengths` *Length of each element within a list*

Description

Get the length of the elements contained in a list.

Usage

```
lengths(alist, out = "vector")
```

Arguments

`alist`	a list
`out`	string indicating the format of the output (`"vector"` or `"list"`)

Value

A vector (or list) with the lengths of the elements in `alist`

Author(s)

Gaston Sanchez

See Also

`length`, `funlist`

Examples

```
# say you have some list
some_list = list(1:3, 4:5, 6:9)

# length of each vector (output in vector format)
lengths(some_list)

# length of each vector (output in list format)
lengths(some_list, out = 'list')

# compare to 'length()'
length(some_list)
```

`listify` *Create a list from a vector of integers*

Description

Given a vector of integers, create a list of indexed vectors.

Usage

```
listify(indices)
```

Arguments

`indices`	a vector of integers indicating the length of each vector in the produced list

Value

A list of index vectors

Author(s)

Gaston Sanchez

See Also

`indexify`

Examples

```
# let's say you have a vector of indices list like this
number_elements = c(3, 1, 5)

# get list of index vectors based on 'number_elements'
listify(number_elements)
```

`listsize` *Size: total number of elements in a list*

Description

Get the total number of elements in a list.

Usage

```
listsize(alist)
```

Arguments

`alist`	a list

Value

number of elements in `alist`.

Author(s)

Gaston Sanchez

See Also

`lengths`

Examples

```
some_list = list(1:3, 4:5, 6:9)

# number of elems in 'some_list'
listsize(some_list)
```

`list_ones` *List with vectors of ones*

Description

Create a list with vectors of ones from a numeric vector

Usage

```
list_ones(x)
```

Arguments

`x`	a numeric vector

Value

A list of vectors with ones

Author(s)

Gaston Sanchez

See Also

`listify`

Examples

```
# let's say you have a numeric vector like this
num_vec = c(2, 3, 1, 4)

# get indices in vector format
list_ones(num_vec)
```

`list_to_dummy` *Create a dummy matrix from the elements in a list*

Description

Create a dummy matrix based on the elements of a list. Each column in the produced matrix is a dummy indicator.

Usage

```
list_to_dummy(alist)
```

Arguments

`alist`	a list of vectors

Value

A matrix of dummy variables

Author(s)

Gaston Sanchez

See Also

`dummy_to_list`, `listify`

Examples

```
# let's say you have a list like this
num_list = list(1:3, 4:5, 6:9)

# get dummy matrix
list_to_dummy(num_list)

# try with a list of strings
str_list = list(c("a","b","c"), c("d", "e"), c("f","g","h"))
list_to_dummy(str_list)
```

`list_to_matrix` *Design-type matrix from the elements in a list*

Description

Create a design-type matrix based on the elements of a list. Each column in the produced matrix is linked to the vectors in the list. See example.

Usage

```
list_to_matrix(alist)
```

Arguments

`alist`	a list of numeric vectors

Value

A design-type matrix

Author(s)

Gaston Sanchez

See Also

`list_to_dummy`, `indexify`

Examples

```
# let's say you have a list like this
num_list = list(1:3, 4:5, 6:9)

# get design-type matrix
list_to_matrix(num_list)
```

`matrix_to_blocks` *Split a matrix into blocks*

Description

Split a matrix into a list of blocks (either by rows or by columns)

Usage

```
matrix_to_blocks(Matrix, blocks, byrow = TRUE)
```

Arguments

`Matrix`	a matrix to split
`blocks`	either a list or a vector indicating the blocks. If `blocks` is a list of vectors, then the length of each vector defines the size of the blocks. If `blocks` is a vector, then each element represents the size of the blocks.
`byrow`	logical. If `TRUE` (the default) the matrix is split by rows, otherwise the matrix is split by columns

Value

A list of matrices

Author(s)

Gaston Sanchez

See Also

`lengths`, `listsize`

Examples

```
# matrix with 10 rows and 7 columns
M = matrix(rnorm(70), 10, 7)

# row blocks
row_sets = list(1:3, 4:5, 6:10)

# split matrix by rows
matrix_to_blocks(M, row_sets)

# column blocks
col_sets = c(3, 4)

# split matrix by rows
matrix_to_blocks(M, col_sets, byrow=FALSE)
```

`maxlist` *Maximum of all elements in a list*

Description

This is just a wrapper of `funlist` using `max`

Usage

```
maxlist(alist, na.rm = FALSE)
```

Arguments

`alist`	a list
`na.rm`	logical indicating whether missing values should be removed

Value

the maximum

Author(s)

Gaston Sanchez

See Also

`funlist`

Examples

```
# say you have some list
list1 = list(1:5, runif(3), rnorm(4))

# get the max of all elements in list1
maxlist(list1)

# say you have missing data
list2 = list(c(1:4, NA), runif(3), rnorm(4))

# get the max of all elements in list2 removing NAs
maxlist(list2, na.rm=TRUE)
```

`meanlist` *Mean of all elements in a list*

Description

This is just a wrapper of `funlist` using `mean`

Usage

```
meanlist(alist, na.rm = FALSE)
```

Arguments

`alist`	a list
`na.rm`	logical indicating whether missing values should be removed

Value

the mean

Author(s)

Gaston Sanchez

See Also

`funlist`

Examples

```
# say you have some list
list1 = list(1:5, runif(3), rnorm(4))

# get the mean of all elements in list1
meanlist(list1)

# say you have missing data
list2 = list(c(1:4, NA), runif(3), rnorm(4))

# get the mean of all elements in list2 removing NAs
meanlist(list2, na.rm=TRUE)
```

`minlist` *Minimum of all elements in a list*

Description

This is just a wrapper of `funlist` using `min`

Usage

```
minlist(alist, na.rm = FALSE)
```

Arguments

`alist`	a list
`na.rm`	logical indicating whether missing values should be removed

Value

the minimum

Author(s)

Gaston Sanchez

See Also

`funlist`

Examples

```
# say you have some list
list1 = list(1:5, runif(3), rnorm(4))

# get the min of all elements in list1
minlist(list1)

# say you have missing data
list2 = list(c(1:4, NA), runif(3), rnorm(4))

# get the min of all elements in list2 removing NAs
minlist(list2, na.rm=TRUE)
```

`prodlist` *Product of all elements in a list*

Description

This is just a wrapper of `funlist` using `prod`

Usage

```
prodlist(alist, na.rm = FALSE)
```

Arguments

`alist`	a list
`na.rm`	logical indicating whether missing values should be removed

Value

the product

Author(s)

Gaston Sanchez

See Also

`funlist`

Examples

```
# say you have some list
list1 = list(1:5, runif(3), rnorm(4))

# get the product of all elements in list1
prodlist(list1)

# say you have missing data
list2 = list(c(1:4, NA), runif(3), rnorm(4))

# get the prod of all elements in list2 removing NAs
prodlist(list2, na.rm=TRUE)
```

`sumlist` *Sum of all elements in a list*

Description

This is just a wrapper of `funlist` using `sum`

Usage

```
sumlist(alist, na.rm = FALSE)
```

Arguments

`alist`	a list
`na.rm`	logical indicating whether missing values should be removed

Value

the sum

Author(s)

Gaston Sanchez

See Also

`funlist`

Examples

```
# say you have some list
list1 = list(1:5, runif(3), rnorm(4))

# get the sum of all elements in list1
sumlist(list1)

# say you have missing data
list2 = list(c(1:4, NA), runif(3), rnorm(4))

# get the sum of all elements in list2 removing NAs
sumlist(list2, na.rm=TRUE)
```

---

turner — **turner** *Turns vectors and lists of vectors into indexed structures*

Description

Package designed for working with vectors and lists of vectors, mainly for turning them into other indexed data structures.

---

vector_to_dummy — *Create a dummy matrix from the elements in a vector*

Description

Create a dummy matrix based on the elements of a vector. Each column in the produced matrix is a dummy indicator.

Usage

```
vector_to_dummy(avector)
```

Arguments

avector	a numeric vector

Value

A matrix of dummy variables

Author(s)

Gaston Sanchez

See Also

list_to_dummy, factor_to_dummy

Examples

```
# let's say you have a list like this
num_vec = c(2, 3, 1, 4)

# get dummy matrix
vector_to_dummy(num_vec)
```

Index