Package ‘matricks’

February 23, 2020

Type Package

Title Useful Tricks for Matrix Manipulation

Version 0.8.2

Description Provides functions, which make matrix creation conciser
(such as the core package’s function m() for rowwise matrix definition or
runifm() for random value matrices).
Allows to set multiple matrix values at once, by using list of formulae.
Provides additional matrix operators and dedicated plotting function.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

BugReports https://github.com/krzjoa/matricks/issues

URL https://github.com/krzjoa/matricks,
https://krzjoa.github.io/matricks/

Suggests testthat (>= 2.1.0), knitr, rmarkdown, covr

RoxygenNote 6.1.1

LinkingTo Rcpp

Imports Rcpp, rlang, ggplot2, reshape2

VignetteBuilder knitr

NeedsCompilation yes

Author Krzysztof Joachimiak [aut, cre]
(<https://orcid.org/0000-0003-4780-7947>)

Maintainer Krzysztof Joachimiak <joachimiak.krzysztof@gmail.com>

Repository CRAN

Date/Publication 2020-02-23 11:40:02 UTC
antidiag

Matrix antidiagonals

Description

Extract or replace the antidiagonal of a matrix, or construct a antidiagonal matrix.

Usage

antidiag(x = as.numeric(c(1)), nrow = NULL, ncol = NULL)

antidiag(x) <- value

Arguments

x matrix, vector or 1D array, or missing.
nrow number of rows (optional; when x is not a matrix)
ncol number of columns (optional; when x is not a matrix)
value either a single value or a vector of length equal to that of the current antidiagonal. Should be of a mode which can be coerced to that of x.
Examples

# Extracting antidiag
antidiag(diag(3))

# Creating antidiagonal matrix
antidiag(7, 3, 3)
antidiag(1:5, 3, 3)

# Assigning antidiagonal
mat <- matrix(0, 3, 3)
antidiag(mat) <- c(3, 4, 5)
mat

```
Examples

# Extracting antidiag
antidiag(diag(3))

# Creating antidiagonal matrix
antidiag(7, 3, 3)
antidiag(1:5, 3, 3)

# Assigning antidiagonal
mat <- matrix(0, 3, 3)
antidiag(mat) <- c(3, 4, 5)
mat

at(mat, idx)
at(mat, idx) <- value

Arguments

mat  matrix
idx  two-element integer vector
value a value to be assign at index

Value

‘at’ function: value from matrix at index idx

Examples

mat <- matrix(0, 3, 3)
idx <- c(1, 2)
# Typically, given matrix and row-column indices as two-element vector, we should do it like this:
mat[idx[1], idx[2]]
mat[idx[1], idx[2]] <- 8
# Using ‘at’, we can do it simpler!
at(mat, idx)
at(mat, idx) <- 7
mat
at(mat, idx)"
binding Bind vector, single values and matrices

Description
This functions works very similar to well-known base `cbind` or `rbind` function. However, there is one big difference between these functions. If you pass a vector, each value will be get individually.

Usage

```r
col_bind(...) # single values, vectors, matrices or data.frames
row_bind(...) # single values, vectors, matrices or data.frames
```

Arguments

... single values, vectors, matrices or data.frames

Value

a matrix being a product of matrix/vector/values binding

Examples

```r
# `col_bind` vs `cbind`
cbind(1,2,3,4,5)
cbind(1,2,3,4,5)
cbind(1:5)
cbind(1:5)
cbind(matrix(3, 3, 3), 0.33, 4:7)
cbind(matrix(3, 3, 3), 0.33, 4:7)
# `row_bind` vs `rbind`
row_bind(1,2,3,4,5)
row_bind(1,2,3,4,5)
row_bind(1:5)
row_bind(1:5)
row_bind(matrix(3, 3, 3), 0.33, 4:7)
row_bind(matrix(3, 3, 3), 0.33, 4:7)
```

is_idx_possible Is idx possible in given matrix?

Description
Is idx possible in given matrix?
Usage

is_idx_possible(mat, idx)

Arguments

mat | matrix
idx | two-element vector

Examples

is_idx_possible(matrix(0, 3, 3), c(4, 5))
is_idx_possible(matrix(0, 3, 3), c(3, 2))

m

A shortcut to create matrix defining rows

Description

One of the main functionalities of the package. It is an alternative to standard way we define matrices in R.

Usage

m(...)

Arguments

... Single values, vectors, matrices and ‘|’ as special symbol which breaks input on the rows.

Value

matrix with defines elements

Examples

# Typically, we define matrices like this:
x <- matrix(c(1, 2, 3,
        4, 5, 6,
        7, 8, 9), nrow=3, byrow=TRUE)
x
# However, this way of ceating matrices seems to be a little bit clunky. Using `matricks`, we can do it in more straightforward way dividing our input into rows by using special symbol `|`
x <- m(1, 2, 3|
        4, 5, 6|
        7, 8, 9)
x # Moreover, we can pass to the `m` function # whole sequences or even matrices.
x <- m(1:5 | 6:10 | 11:15 )
x # We can combine multiple matrices into one
m(diag(3), diag(3) * 3 |
   diag(3) * 3, diag(3) )

matrix_idx

Get available marix indices

Description

Get available marix indices

Usage

matrix_idx(mat, n.row = NULL, n.col = NULL, mask = NULL)

Arguments

mat    matrix
n.row  number of rows; default: NULL
n.col  number of columns; default: NULL
mask   logical matrix; default: NULL

Examples

T <- TRUE; F <- FALSE
mat <- matrix(0, 3, 3)
mask <- m(T, T, F | T, F, T | F, F, T)
# All poss
matrix_idx(mat)
matrix_idx(mat, mask = mask)
matrix_idx(mask = mask)
neighbour_idx

Get all indices in neighbourhood

Description
Get all indices in neighbourhood

Usage

\[
\text{neighbour_idx(mat, idx, mask = NULL, diagonal = TRUE,}
\text{ include.idx = FALSE)}
\]

Arguments

- \text{mat}: matrix or data.frame
- \text{idx}: two-element vector
- \text{mask}: logical matrix; optional
- \text{diagonal}: include diagonal neighbours
- \text{include.idx}: include current index

Examples

\[
\begin{align*}
\text{mat} & \leftarrow \text{matrix}(0, 3, 3) \\
\text{neighbour_idx(mat, c(1, 2))} \\
\text{neighbour_idx(mat, c(1, 2), diagonal = FALSE)} \\
\text{neighbour_idx(mat, c(1, 2), diagonal = FALSE, include.idx = TRUE)} \\
\text{# With mask} \\
\text{mat} & \leftarrow \text{matrix}(0, 3, 4) \\
\text{mask} & \leftarrow \text{m}(\text{FALSE, FALSE, TRUE, TRUE} | \\
\text{FALSE, FALSE, FALSE, FALSE} | \\
\text{TRUE, TRUE, FALSE, TRUE}) \\
\text{neighbour_idx(mat, c(1, 2), mask = mask)}
\end{align*}
\]

neighbour_idx_matrix

Create matrix of lists, where each one contains list of neighbour field coordinates

Description
Create matrix of lists, where each one contains list of neighbour field coordinates

Usage

\[
\text{neighbour_idx_matrix(mat, mask = NULL, diagonal = TRUE,}
\text{ random.select = NULL)}
\]
Arguments

mat matrix
mask logical matrix. Its dimensions must be identical with dimensions of mat
diagonal logical. get diagonal neighbours
random.select select one random neighbour

Examples

T <- TRUE; F <- FALSE
mat <- matrix(0, 3, 3)
mask <- m(T, T, F | T, F, T | F, F, T)
nimat <- neighbour_idx_matrix(mat, mask, diagonal = TRUE)
neighbour_idx_matrix(mat, mask, diagonal = TRUE, random.select = 1)

operators Binary operations on matrices/vectors

Description

This operator allows to do elementwise operation of two algebraic object i.e. matrices/vectors.
There is one required condition to perform such operation: at least one dimension values from both
objects must be the same

Usage

a %m% b
a %d% b
a %-% b
a %+% b

Arguments

a matrix/vector
b matrix/vector

Value

Matrix/vector
### plot_matrix

#### Plot a matrix

This function allows us to plot matrices easily.

#### Usage

```r
plot_matrix(x, ...)  
```  

#### Arguments

- `x`: a matrix
- `...`: for S3 generic API consistency; does nothing

#### Value

A ggplot object

#### Examples

```r
T <- TRUE; F <- FALSE  
x1 <- m(T, T, T, F, T |  
     T, T, F, T, T |  
     F, T, T, F |  
     T, T, T, T |  
     F, F, T, T |  
     F, T, T, F)  
plot_matrix(x1)  
x2 <- m(T, T, F, T |  
     T, T, F, T |  
     T, F, T, T |  
     T, T, T, T |  
     T, F, T, F)  
plot(x2)  
x3 <- m(runif(3) | runif(3) | runif(3))  
plot(x3)
```
rboolm  

Create matrix of random chosen boolean values

Description

Create matrix of random chosen boolean values

Usage

rboolm(nrow, ncol, true.proba = 0.5)

Arguments

nrow  number of rows
ncol  number of columns
true.proba  probability of true values; default: 0.5

Value

a matrix

Examples

rboolm(3, 3)
rboolm(4, 5, true.proba = 0.3)

repetitions  

Repeat columns or rows

Description

Repeat matrix object respectively to its shape and orientation

Usage

crep(x, times)
rrep(x, times)

Arguments

x  matrix
times  number of repetitions
runifm

Details

crep = columnwise repetition
rrep = rowwise repetition

Value

matrix

Examples

# Columnwise repetition
crep(v(1:3), 4)
crep(t(v(1:5)), 4)
# Rowwise repetition
rrep(v(1:3), 4)
rrep(t(v(1:5)), 4)

Description

Create matrix of random values drawn from uniform distribution

Usage

runifm(nrow, ncol, min = 0, max = 1)

Arguments

nrow number of rows
ncol number of columns
min lower limit of the distribution. Must be finite.
max upper limit of the distribution. Must be finite.

Value

a matrix

Examples

runifm(3, 3)
runifm(4, 5, min = -1, max = 3)
runif_same_dims  

Create matrix of random values with dimensions copied from an existing matrix

Description

Create matrix of random values with dimensions copied from an existing matrix

Usage

runif_same_dims(mat, min = 0, max = 1)

Arguments

mat  
matrix

min  
lower limit of the distribution. Must be finite.

max  
upper limit of the distribution. Must be finite.

Value

a matrix

Examples

mat <- matrix(0, 3, 3)
runif_same_dims(mat)

seq_matrix

Return a sequence of pairs (value, index vector)

Description

Facilitates iterating over matrix, returning a sequence of pairs, where the first element is a value at index (x, y) and the second one is the index (x, y)

Usage

seq_matrix(mat)

Arguments

mat  
matrix

Value

list of two-element list (single value, two-element vector)
**Examples**

```r
mat <- matrix(1:9, 3, 3)
seq_matrix(mat)
```

---

**set_values**

*Set multiple values using one function call*

**Description**

This function allows to set multiple elements of a matrix instead of using annoying step-by-step assignment by `mat[1,2] <- 2 mat[2,3] <- 0.5` etc.

**Usage**

```r
set_values(mat, ...)
sv(mat, ...)
```

**Arguments**

- `mat`: a matrix object
- `...`: formulae; left hand values should be two-element integer vectors and right-hand: a single-value numeric

**Value**

matrix

**Examples**

```r
mat <- matrix(0, 4, 5)
set_values(mat, c(1,1) ~ 5, c(3, 4) ~ 0.3)
```

---

**v**

*A shortcut to create a vertical vector*

**Description**

This function provides convenient shortcut to create a vertical (column) vector.

**Usage**

`v(...)`
with_same_dims

Arguments

... arbitrary number of values

Value

matrix with dims n_elements x 1

Examples

# Enumerating all the values with commas
v(1, 2, 3)
# Passing whole sequence as an argument
v(1:5)

with_same_dims(mat, data)

Arguments

mat a matrix with desired dimensions
data sigle numeric value or numeric vector

Value

a matrix

Examples

x <- matrix(7, 3, 6)
x
with_same_dims(x, 0)
with_same_dims(x, c(1, 2))
## Index

<table>
<thead>
<tr>
<th>Index</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>%+% (operators)</td>
<td>8</td>
</tr>
<tr>
<td>%-% (operators)</td>
<td>8</td>
</tr>
<tr>
<td>%d% (operators)</td>
<td>8</td>
</tr>
<tr>
<td>%m% (operators)</td>
<td>8</td>
</tr>
<tr>
<td>antidiag</td>
<td>2</td>
</tr>
<tr>
<td>antidiag&lt;- (antidiag)</td>
<td>2</td>
</tr>
<tr>
<td>at</td>
<td>3</td>
</tr>
<tr>
<td>at&lt;- (at)</td>
<td>3</td>
</tr>
<tr>
<td>binding</td>
<td>4</td>
</tr>
<tr>
<td>col_bind (binding)</td>
<td>4</td>
</tr>
<tr>
<td>crep (repetitions)</td>
<td>10</td>
</tr>
<tr>
<td>is_idx_possible</td>
<td>4</td>
</tr>
<tr>
<td>m</td>
<td>5</td>
</tr>
<tr>
<td>matrix_idx</td>
<td>6</td>
</tr>
<tr>
<td>neighbour_idx</td>
<td>7</td>
</tr>
<tr>
<td>neighbour_idx_matrix</td>
<td>7</td>
</tr>
<tr>
<td>operators</td>
<td>8</td>
</tr>
<tr>
<td>plot.matrix (plot_matrix)</td>
<td>9</td>
</tr>
<tr>
<td>plot_matrix</td>
<td>9</td>
</tr>
<tr>
<td>rboolm</td>
<td>10</td>
</tr>
<tr>
<td>repetitions</td>
<td>10</td>
</tr>
<tr>
<td>row_bind (binding)</td>
<td>4</td>
</tr>
<tr>
<td>rrep (repetitions)</td>
<td>10</td>
</tr>
<tr>
<td>runif_same_dims</td>
<td>12</td>
</tr>
<tr>
<td>runifm</td>
<td>11</td>
</tr>
<tr>
<td>seq_matrix</td>
<td>12</td>
</tr>
<tr>
<td>set_values</td>
<td>13</td>
</tr>
<tr>
<td>sv (set_values)</td>
<td>13</td>
</tr>
<tr>
<td>v</td>
<td>13</td>
</tr>
<tr>
<td>with_same_dims</td>
<td>14</td>
</tr>
</tbody>
</table>