Package ‘matricks’

October 13, 2022

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
R topics documented:

antidiag ................................................................. 2
at ................................................................. 3
binding ................................................................. 4
is_idx_possible ................................................................. 4
m ................................................................. 5
matrix_idx ................................................................. 6
neighbour_idx ................................................................. 7
neighbour_idx_matrix ................................................................. 7
operators ................................................................. 8
plot_matrix ................................................................. 9
rboolm ................................................................. 10
repetitions ................................................................. 10
runifm ................................................................. 11
runif_same_dims ................................................................. 12
seq_matrix ................................................................. 12
set_values ................................................................. 13
v ................................................................. 13
with_same_dims ................................................................. 14

Index

15

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

at                                 Set or get matrix value at index vector

Description

This function allows to access matrix values by passing indices as vector

Usage

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 simplier!
at(mat, idx)
at(mat, idx) <- 7
mat
at(mat, idx)
binding

**Bind vector, single values and matrices**

**Description**

This function 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(...)  
row_bind(...)  
```

**Arguments**

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

```r
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

neighbour_idx(mat, idx, mask = NULL, diagonal = TRUE, include.idx = FALSE)

Arguments

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

Examples

mat <- matrix(0, 3, 3)
neighbour_idx(mat, c(1, 2))
neighbour_idx(mat, c(1, 2), diagonal = FALSE)
neighbour_idx(mat, c(1, 2), diagonal = FALSE, include.idx = TRUE)
# With mask
mat <- matrix(0, 3, 4)
mask <- m(FALSE, FALSE, TRUE, TRUE |
           FALSE, FALSE, FALSE, FALSE |
           TRUE, TRUE, FALSE, TRUE)
neighbour_idx(mat, c(1, 2), mask = mask)

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

neighbour_idx_matrix(mat, mask = NULL, diagonal = TRUE, 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

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
Examples

# Multiply
m(1, 2, 3 | 4, 5, 6 | 7, 8, 9) %m% v(5, 4, 3)
# Divide
m(1, 2, 3 | 4, 5, 6 | 7, 8, 9) %d% v(5, 4, 3)
# Add
m(1, 2, 3 | 4, 5, 6 | 7, 8, 9) +%% v(5, 4, 3)
# Subtract
m(1, 2, 3 | 4, 5, 6 | 7, 8, 9) %-% v(5, 4, 3)

plot_matrix

Description
This function allows us to plot matrices easily

Usage
plot_matrix(x, ...)

## S3 method for class 'matrix'
plot(x, ...)

Arguments

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

Value
a ggplot object

Examples
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 )
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
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)

---

runifm Create matrix of random values drawn from uniform distribution

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mat</td>
<td>matrix</td>
</tr>
<tr>
<td>min</td>
<td>lower limit of the distribution. Must be finite.</td>
</tr>
<tr>
<td>max</td>
<td>upper limit of the distribution. Must be finite.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mat</td>
<td>matrix</td>
</tr>
</tbody>
</table>

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

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

---

**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**

```
set_values(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**

```
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(...)```
**Arguments**

... arbitrary number of values

**Value**

matrix with dims n_elements x 1

**Examples**

```r
# Enumerating all the values with commas
v(1, 2, 3)

# Passing whole sequence as an argument
v(1:5)
```

---

**with_same_dims** Create new matrix copying dimensions from the existing one

**Description**

Create new matrix copying dimensions from the existing one

**Usage**

```r
with_same_dims(mat, data)
```

**Arguments**

- `mat` a matrix with desired dimensions
- `data` single numeric value or numeric vector

**Value**

a matrix

**Examples**

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

%+% (operators), 8
%-% (operators), 8
%d% (operators), 8
%m% (operators), 8

antidiag, 2
antidiag<- (antidiag), 2
at, 3
at<- (at), 3

binding, 4
col_bind (binding), 4
crep (repetitions), 10

is_idx_possible, 4

m, 5
matrix_idx, 6

neighbour_idx, 7
neighbour_idx_matrix, 7

operators, 8

plot.matrix (plot_matrix), 9
plot_matrix, 9

rboolm, 10
repetitions, 10
row_bind (binding), 4
rrep (repetitions), 10
runif_same_dims, 12
runifm, 11

seq_matrix, 12
set_values, 13
sv (set_values), 13

v, 13

with_same_dims, 14