Package ‘ramify’

December 17, 2016

Type  Package
Title  Additional Matrix Functionality
Version 0.3.3
Description Additional matrix functionality for R including: (1) wrappers for the base matrix function that allow matrices to be created from character strings and lists (the former is especially useful for creating block matrices), (2) better printing of large matrices via the generic “pretty” print function, and (3) a number of convenience functions for users more familiar with other scientific languages like 'Julia', 'Matlab'/Octave', or 'Python'+'NumPy'.

Imports  stats
Suggests testthat, knitr
License GPL (>= 2)
URL https://github.com/bgreenwell/ramify
BugReports https://github.com/bgreenwell/ramify/issues
VignetteBuilder knitr
RoxygenNote 5.0.1
NeedsCompilation no
Author Brandon Greenwell [aut, cre]
Maintainer Brandon Greenwell <greenwell.bradon@gmail.com>
Repository CRAN
Date/Publication 2016-12-17 22:46:24

R topics documented:

argmax ........................................ 2
atleast_2d .................................... 3
bmat ........................................... 3
clip ............................................ 4
dmat .......................................... 5
argmax

Row/Column Max/Min Indices

Description

Returns the indices of the maximum or minimum values along an axis.

Usage

argmax(x, rows = TRUE)

argmin(x, rows = TRUE)

Arguments

x
A matrix.

rows
If TRUE (the default) the indices of each row max/min is returned.

Value

A vector of indices.
atleast_2d

Description

Ensure that the input has at least two dimensions.

Usage

atleast_2d(x)

Arguments

x An R object, for example a vector, matrix, array, or data frame.

Value

The same object, but with a "dim" attribute.

Examples

x <- 1:10
x
atleast_2d(x)

bmat

Block Matrices

Description

Construct a block matrix using a character string initializer.

Usage

bmat(x, rows = TRUE, sep = ",", ...)
clip

Arguments

- **x**: A data vector, character string, or a list.
- **rows**: Logical. If TRUE (the default) the matrix is filled by rows, otherwise the matrix is filled by columns.
- **sep**: Separator string. Values within each row/column of `x` are separated by this string. Default is ",".
- **...**: Additional optional arguments.

Value

A matrix (i.e., an object of class "matrix").

See Also

`mat`, `dmat`.

Examples

```r
# Construct a block matrix from matrices A1, A2, and A3
A1 <- mat('1, 1; 1, 1')
A2 <- mat('2, 2; 2, 2')
A3 <- mat('3, 3, 3; 3, 3, 3')
pmat('A1, A2, A3')
```

Description

Clip (i.e., limit) the values in a vector, matrix, or array.

Usage

```r
clip(x, .min, .max, ...)
```

## Default S3 method:

```r
clip(x, .min, .max, ...)
```

Arguments

- **x**: A vector, matrix, or multi-way array.
- **.min**: Minimum value.
- **.max**: Maximum value.
- **...**: Additional optional arguments.
Value

Returns x with values outside the interval \([\text{min}, \text{max}]\) clipped to the interval edges. That is, values in \(x\) smaller than \(\text{min}\) become \(\text{min}\), and values larger than \(\text{max}\) become \(\text{max}\).

Examples

\[
\text{clip}(1:10, 3, 8) \quad \# [1] 3 3 3 4 5 6 7 8 8 8
\]
\[
\text{clip(\text{randa}(5, 5), \text{.min} = -1, \text{.max} = 1)}
\]

---

dmat

Data Frames

Description

Like \text{mat}, but returns a data frame.

Usage

\[
dmat(x, \ldots)
\]

Arguments

\[
x \quad \text{A data vector, character string, or a list.}
\]
\[
\ldots \quad \text{Additional optional arguments passed on to \text{mat}.}
\]

Value

A data frame (i.e., an object of class "\text{data.frame}").

See Also

\text{mat}, \text{bmat}.

Examples

\[
dmat(\text{\'1e-01, 2+5, 3, 4, 5; 6, 7, 8, 9^2, \text{pi}'}, \text{rows = FALSE})
\]
\[
z \leftarrow \text{\text{list(a = 1:10, b = 11:20, c = 21:30)}
\]
\[
dmat(z) \quad \# \text{list elements form rows}
\]
\[
dmat(z, \text{rows= FALSE}) \quad \# \text{list elements form columns}
\]
**eye**  
*Identity Matrix*

**Description**

Creates an `nrow`-by-`ncol` identity matrix.

**Usage**

```r
eye(nrow = 1, ncol = nrow)
```

**Arguments**

- `nrow`  
The desired number of rows.
- `ncol`  
The desired number of columns.

**Value**

A `nrow`-by-`ncol` identity matrix.

**See Also**

`diag`

**Examples**

```r
eye(4)  # 4-by-4 identity matrix
eye(4, 4)  # 4-by-4 identity matrix
eye(3, 5)  # 3-by-5 identity matrix
eye(5, 3)  # 5-by-3 identity matrix
```

---

**fill**  
*Fill a Matrix*

**Description**

Create a matrix filled with the value `x`. 
## flatten

### Usage

```r
fill(x, nrow = 1, ncol = 1, ..., atleast_2d = NULL)
false(nrow = 1, ncol = 1, ..., atleast_2d = NULL)
true(nrow = 1, ncol = 1, ..., atleast_2d = NULL)
ones(nrow = 1, ncol = 1, ..., atleast_2d = NULL)
zeros(nrow = 1, ncol = 1, ..., atleast_2d = NULL)
```

### Arguments

- **x**: The (single) value to fill the matrix with.
- **nrow**: The desired number of rows.
- **ncol**: The desired number of columns.
- **...**: Further dimensions of the array.
- **atleast_2d**: Logical indicating whether or not to force column vectors to have a second dimension equal to one. Defaults to `false`. This behavior can also be changed globally using, for example `options(atleast_2d = TRUE)`.

### Value

A matrix or array filled with the value `x`.

### See Also

`ones`, `zeros`, `false`, `true`, `mat`, `matrix`.

### Examples

```r
fll(pi, 3, 5) # 3-by-5 matrix filled with the value of pi
fll(pi, 3, 5, 2, 2) # 3-by-5-by-2-by-2 array filled with the value of pi
pi * ones(3, 5)
zeros(10)
zeros(10, atleast_2d = TRUE)
```

---

## flatten

### Description

Flatten (i.e., collapse) a matrix or array to one dimension.

### Usage

```r
flatten(x, across = c("rows", "columns"))
```
Arguments

x  A matrix object.
across  Character string specifying whether to flatten the matrix across "rows" (default) or "columns". This option is ignored for multi-way arrays.

Value

A numeric vector.

See Also

mat.

Examples

m <- mat("2, 4, 6, 8; 10, 12, 14, 16")
flatten(m)
flatten(m, across = "columns")

Description

Concatenate matrices along the first or second dimension.

Usage

hcat(...)  vcat(...)

Arguments

...  Vectors or matrices.

Value

A matrix formed by combining the ... arguments column-wise (hcat) or row-wise (vcat).

See Also

bmat, cbind, rbind.

Examples

m1 <- mat("1, 2, 3; 4, 5, 6")
m2 <- mat("7, 8, 9; 10, 11, 12")
hcat(m1, m2)  # same as 'bmat("m1, m2")'
vcat(m1, m2)  # same as 'bmat("m1; m2")'
inv  Matrix Inverse

Description
Calculates the inverse of a square matrix.

Usage
inv(x, ...)

Arguments
x  A square numeric or complex matrix
...
Additional optional arguments.

Details
See the documentation for the base function solve.

See Also
solve.

Examples
m <- 3 * eye(5)
inv(m)

is.tril  Lower Triangular Matrix Test

Description
Determine if a Matrix is Lower Triangular

Usage
is.tril(x)

Arguments
x  A matrix

Value
Logical indicating whether the given matrix is lower triangular.
Examples

\[
\begin{align*}
    m &\leftarrow \text{mat(}^{1, 0, 0, 0; -1, 1, 0, 0; -2, -2, 1, 0; -3, -3, -3, 1}^{M}) \\
    \text{is.tril}(m) & \\
    \text{is.tril}(\text{eye}(3, 5)) \\
\end{align*}
\]

is.triu \hspace{1cm} \textit{Upper Triangular Matrix Test}

Description

Determine if a Matrix is Upper Triangular

Usage

\[
\text{is.triu}(x)
\]

Arguments

\[
\begin{align*}
    x & \hspace{1cm} \text{A matrix}
\end{align*}
\]

Value

Logical indicating whether the given matrix is lower triangular.

Examples

\[
\begin{align*}
    m &\leftarrow \text{mat(}^{1, -1, -1, -1; 0, 1, -2, -2; 0, 0, 1, -3; 0, 0, 0, 1}^{M}) \\
    \text{is.triu}(m) & \\
    \text{is.triu}(\text{eye}(3, 5)) \\
\end{align*}
\]

linspace \hspace{1cm} \textit{Linearly-spaced Elements}

Description

Construct a vector of \( n \) linearly-spaced elements from \( a \) to \( b \).

Usage

\[
\text{linspace}(a, b, n = 50)
\]

Arguments

\[
\begin{align*}
    a & \hspace{1cm} \text{The starting value of the sequence.} \\
    b & \hspace{1cm} \text{The final value of the sequence.} \\
    n & \hspace{1cm} \text{The number of samples to generate. Default is 50.}
\end{align*}
\]
logspace

Value

A vector of linearly-spaced elements.

See Also

logspace, seq.

Examples

```matlab
linspace(0, 1)
linspace(1, 5, 5)
linspace(1+2i, 10+10i, 8)
logspace(0, pi, 10)
```

Description

Construct a vector of \( n \) logarithmically-spaced elements from \( 10^a \) to \( 10^b \).

Usage

```matlab
logspace(a, b, n = 50, base = 10)
```

Arguments

- \( a \) base\(^a\) is the starting value of the sequence.
- \( b \) base\(^b\) is the final value of the sequence.
- \( n \) The number of samples to generate. Default is 50.
- base The base of the log space.

Value

A vector of logarithmically-spaced elements.

Note

If \( b = \pi \) and base = 10, the points are between \( 10^a \) and \( \pi \), not \( 10^a \) and \( 10^\pi \), for compatibility with the corresponding MATLAB/Octave, and NumPy functions.

See Also

linspace, seq.
Description

Like `matrix`, `mat` creates a matrix from the given set of values. However, these values can also be represented by a character string, or a list of vectors. Initially inspired by NumPy’s `matrix` function.

Usage

```r
mat(x, ...)
```

```r
## Default S3 method:
mat(x, ...)
```

```r
## S3 method for class 'character'
mat(x, rows = TRUE, sep = " ", eval = FALSE, ...)
```

```r
## S3 method for class 'list'
mat(x, rows = TRUE, ...)
```

Arguments

- `x` A data vector, character string, or a list.
- `...` Additional optional arguments to be passed on to `matrix`.
- `rows` Logical. If `TRUE` (the default) the matrix is filled by rows, otherwise the matrix is filled by columns.
- `sep` Separator string. Values within each row/column of `x` are separated by this string. Default is `" ", " `. 
- `eval` Logical indicating whether or not the character string contains R expressions that need to be evaluated. Default is `FALSE`. See examples below for usage.

Value

A matrix (i.e., an object of class "matrix").

See Also

`bmat`, `dmatrix`, `matrix`.

Examples

```r
## Creating a matrix from a character string
mat("1, 2, 3, 4; 5, 6, 7, 8") # ";" separates rows
mat("1, 2, 3, 4; 5, 6, 7, 8", rows = FALSE) # ";" separates columns
mat("1 2 3 4; 5 6 7 8", sep = " ") # use spaces instead of commas
mat(c(1, 2, 3, 4, 5, 6, 7, 8), nrow = 2, byrow = TRUE) # works like matrix too
```
# Character strings containing R expressions
mat("rnorm(3); rnorm(3)")
mat("rnorm(3); rnorm(3)", eval = TRUE)
mat("1, 2, 3; 4, 5, pi")
mat("1, 2, 3; 4, 5, pi", eval = TRUE)

# Creating a matrix from a list
z1 <- list(1:5, 6:10)
z2 <- list(a = 1:5, b = 6:10)
mat(z1)
mat(z2) # preserves names as row names
mat(z2, rows = FALSE) # preserves names as column names

---

**matrix_rank**

**Matrix Rank**

**Description**

Compute the rank of a matrix using the singular value decomposition (SVD) method.

**Usage**

```
matrix_rank(x, tol)
```

```r
## Default S3 method:
matrix_rank(x, tol)
```

```r
## S3 method for class 'matrix'
matrix_rank(x, tol)
```

```r
## S3 method for class 'data.frame'
matrix_rank(x, tol)
```

**Arguments**

- `x` an object that inherits from class "matrix".
- `tol` Threshold below which SVD values are considered zero.

**Details**

The singular value decomposition method simply computes the SVD of `x` and returns the number of singular values of `x` that are greater than `tol`. See the function `rankMatrix` in package `Matrix` for alternative methods.
Examples

matrix_rank(1:5)
matrix_rank(randn(2, 2))
matrix_rank(cbind(c(1, 1, 1), c(2, 2, 2)))
matrix_rank(ones(3, 3))
matrix_rank(zeros(3, 5))

---

meshgrid

Rectangular 2-D Grid

Description

Creates matrices for vectorized evaluations of 2-D scalar/vector fields over 2-D grids.

Usage

meshgrid(x, y = x)

Arguments

x Numeric vector representing the first coordinate of the grid.

y Numeric vector representing the second coordinate of the grid.

Value

A list of matrices.

See Also

expand.grid, outer.

Examples

mg <- meshgrid(linspace(-4*pi, 4*pi, 27)) # list of input matrices
z <- cos(mg[[1]]^2 + mg[[2]]^2) * exp(-sqrt(mg[[1]]^2 + mg[[2]]^2)/2)/6
image(z, axes = FALSE) # color image
contour(z, add = TRUE, drawlabels = FALSE) # add contour lines
Description

Prettier printing for matrices and data frames.

Usage

pprint(x, ...)

## S3 method for class 'matrix'
pprint(x, rowdots = NULL, coldots = NULL, digits = NULL, ...)

## S3 method for class 'data.frame'
pprint(x, rowdots = NULL, coldots = NULL, digits = NULL, ...)

Arguments

x An object of class "matrix" or "data.frame".

... Additional optional arguments. None are used at present.

rowdots Integer specifying the row to replace with \ldots notation. Default is 4.

coldots Integer specifying the column to replace with \ldots notation. Default is 4.

digits The minimum number of significant digits to be printed in values.

Details

For object of class "matrix" or "data.frame" (which are coerced to a matrix via the data.matrix function), pprint will replace all the rows starting from rowdots up to and including the second-to-last row with a single row filled with \ldots. The same is applied to the columns as well. Hence a large matrix (or data frame) will be printed in a much more compact form.

Examples

pprint(randn(100, 100))
pprint(resize(1:100, 10, 10))
Description

Additional matrix functionality for R including: (1) wrappers for the base matrix function that allows matrices to be created from character strings and lists (the former is especially useful for creating block matrices), (ii) better printing of large matrices via a new generic function for "pretty" printing, and (iii) a number of convenience functions for users more familiar with other scientific languages like ‘Julia’, ‘Matlab’/’Octave’, or ‘Python’+’NumPy’.

Details

To learn more about ramify, read the introductory vignette: browseVignettes(package = "ramify")

Description

Construct a matrix or multi-way array of uniform random deviates.

Usage

```r
rand(nrow = 1L, ncol = 1L, ..., min = 0L, max = 1L, atleast_2d = NULL)
```

Arguments

- `nrow` The desired number of rows.
- `ncol` The desired number of columns.
- `...` Further dimensions of the array.
- `min` Lower limit for the uniform distribution. Must be finite. (rand only).
- `max` Upper limit for the uniform distribution. Must be finite. (rand only).
- `atleast_2d` Logical indicating whether or not to force column vectors to have a second dimension equal to one. Defaults to FALSE. This behavior can also be changed globally using, for example options(atleast_2d = TRUE).

Value

A matrix or array of pseudorandom numbers.

See Also

`randi`, `randn`, `runif`. 
**randi**

**Examples**

```r
randi(100, 100)  # 100 by 100 matrix of uniform random numbers
rand(2, 3, min = 100, max = 200)
```

---

**Matrix/Array of Uniform Random Integers**

**Description**

Construct a matrix or multi-way array of uniform random integers.

**Usage**

```r
randi(imax, nrow, ncol = 1, ..., atleast_2d = NULL)
```

**Arguments**

- **imax**: A positive integer.
- **nrow**: The desired number of rows.
- **ncol**: The desired number of columns.
- **...**: Further dimensions of the array.
- **atleast_2d**: Logical indicating whether or not to force column vectors to have a second dimension equal to one. Defaults to `FALSE`. This behavior can also be changed globally using, for example `options(atleast_2d = TRUE)`.

**Value**

A matrix or array of pseudorandom numbers.

**See Also**

- `rand`, `randn`, `sample`.

**Examples**

```r
randi(2, 5, 5)
```
**Matrix/Array of Normal Random Numbers**

**Description**
Construct a matrix or multi-way array of normal random deviates.

**Usage**
```
randn(nrow = 1, ncol = 1, ..., mean = 0, sd = 1, atleast_2d = NULL)
```

**Arguments**
- `nrow` The desired number of rows.
- `ncol` The desired number of columns.
- `...` Further dimensions of the array.
- `mean` Mean for the normal distribution. (``randn`` only).
- `sd` Standard deviation for the normal distribution. (``randn`` only).
- `atleast_2d` Logical indicating whether or not to force column vectors to have a second dimension equal to one. Defaults to `FALSE`. This behavior can also be changed globally using, for example `options(atleast_2d = TRUE)`.

**Value**
A matrix or array of pseudorandom numbers.

**See Also**
```
rand, randi, rnorm.
```

**Examples**
```
randn(100, 100) # 100 by 100 matrix of standard normal random variates
randn(2, 3, mean = 10, sd = 0.1)
```
Repeat Vectors and Matrices

Description

Repeat a vector or matrix a specific number of times.

Usage

repmat(x, m, n)

Arguments

x
A vector or matrix.
m
Integer specifying how many times to repeat x in the first dimension.
n
Integer specifying how many times to repeat x in the second dimension.

Value

A block matrix of dimension m*row(x) by n*ncol(x).

Examples

repmat(1:3, 3, 2)  # will have dimension 9 by 2
repmat(randn(2, 2), 3, 2)

Resize Matrices and Arrays

Description

Change shape and size of a matrix or array.

Usage

resize(x, nrow, ncol, ..., across = c("rows", "columns"), byrow = FALSE)

Arguments

x
A matrix or multi-way array.
nrow
The desired number of rows.
ncol
The desired number of columns.
... Further dimensions of the array.
across Character string specifying whether to flatten the matrix across "rows" (default) or "columns". This option is ignored for multi-way arrays.
byrow Logical. If FALSE (default) the new matrix is filled by columns, otherwise it is filled by rows. This option is ignored for multi-way arrays.
Value

A matrix with dimension nrow-by-ncol.

See Also

flatten, mat, matrix.

Examples

```r
m <- 1:9
resize(m)
resize(m, 3, 3)
resize(m, 2, 2)
```

Description

Retrieve the dimensions of a matrix or array.

Usage

```r
size(x)
```

Arguments

- `x` A matrix, array, or data frame.

Value

The dimensions of the object.

See Also

- `dim`

Examples

```r
m <- mat("1, 3, 5; 7, 9, 11")
size(m)
```
**tr**  
*Trace of a Matrix*

**Description**
Sum of diagonal elements of a matrix.

**Usage**
\[
\text{tr}(x)
\]

**Arguments**
- \(x\) A matrix.

**Value**
The sum of the diagonal elements of \(x\).

**Examples**
\[
\begin{align*}
\text{tr(ones(5, 10))} \\
x & \leftarrow \text{replicate(1000, tr(rand(25, 25)))} \\
\text{hist}(x)
\end{align*}
\]

---

**tri**  
*Lower/Upper Triangular Matrix*

**Description**
Construct a matrix with ones at and below the given diagonal and zeros elsewhere.

**Usage**
\[
\text{tri}(\text{nrow}, \text{ncol} = \text{nrow}, k = 0, \text{diag} = \text{TRUE})
\]

**Arguments**
- \(\text{nrow}\) The desired number of rows.
- \(\text{ncol}\) The desired number of columns.
- \(k\) The sub-diagonal at and below which the matrix is filled. \(k = 0\) is the main diagonal, while \(k < 0\) is below it, and \(k > 0\) is above. The default is 0.
- \(\text{diag}\) Logical indicating whether to include the diagonal. Default is \text{TRUE}. 

Examples

\begin{verbatim}
tri(5, 5)
tri(5, 5, 2)
tri(5, 5, -1)
\end{verbatim}

\section*{tril}

\textit{Extract Lower Triangular Matrix}

\section*{Description}

Extract the lower triangular part of a matrix.

\section*{Usage}

\begin{verbatim}
tril(x, k = 0, diag = TRUE)
\end{verbatim}

\section*{Arguments}

\begin{itemize}
\item \textit{x} \hspace{1cm} A matrix.
\item \textit{k} \hspace{1cm} Diagonal above which to zero elements. \textit{k} = 0 (the default) is the main diagonal, \textit{k} < 0 is below it and \textit{k} > 0 is above.
\item \textit{diag} \hspace{1cm} Logical indicating whether to include the diagonal. Default is TRUE.
\end{itemize}

\section*{Examples}

\begin{verbatim}
tril(ones(5, 5))
tril(ones(5, 5), diag = TRUE)
\end{verbatim}

\section*{triu}

\textit{Extract Upper Triangular Matrix}

\section*{Description}

Extract the upper triangular part of a matrix.

\section*{Usage}

\begin{verbatim}
triu(x, k = 0, diag = TRUE)
\end{verbatim}

\section*{Arguments}

\begin{itemize}
\item \textit{x} \hspace{1cm} A matrix.
\item \textit{k} \hspace{1cm} Diagonal below which to zero elements. \textit{k} = 0 (the default) is the main diagonal, \textit{k} < 0 is below it and \textit{k} > 0 is above.
\item \textit{diag} \hspace{1cm} Logical indicating whether to include the diagonal. Default is TRUE.
\end{itemize}
Examples

```
triu(ones(5, 5))
triu(ones(5, 5), diag = FALSE)
```
Index

argmax, 2
argmin (argmax), 2
atleast_2d, 3

bmat, 3, 5, 8, 12

cbind, 8
clip, 4
diag, 6
dim, 20
dmat, 4, 5, 12

expand.grid, 14
eye, 6

cbind, 8
false, 7
false (fill), 6
fill, 6
flatten, 7, 20

hcat, 8
inv, 9
is.tril, 9
is.triu, 10
linspace, 10, 11
logspace, 11, 11

mat, 4, 5, 7, 8, 12, 20
Matrix, 13
matrix, 7, 12, 20
matrix_rank, 13
meshgrid, 14

ones, 7
ones (fill), 6
outer, 14

pprint, 15

ramify, 16
ramify-package (ramify), 16
rand, 16, 17, 18
randi, 16, 17, 18
randn, 16, 17, 18
rankMatrix, 13
rbind, 8
repmat, 19
resize, 19
rnorm, 18
runif, 16

sample, 17
seq, 11
size, 20
solve, 9

tr, 21
tri, 21
tril, 22
triu, 22
trues, 7
trues (fill), 6

vcat (hcat), 8

zeros, 7
zeros (fill), 6