Package ‘matlab2r’

May 13, 2022

Title Translation Layer from MATLAB to R
Version 1.1.1
Description Allows users familiar with MATLAB to use MATLAB-named functions in R. Several basic MATLAB functions are written in this package to mimic the behavior of their original counterparts, with more to come as this package grows.
License GPL (>= 3)
Encoding UTF-8
RoxygenNote 7.1.2
Suggests testthat (>= 3.0.0)
Config/testthat/edition 3
Imports methods, styler
Date 2022-05-13
NeedsCompilation no
Author Waldir Leoncio [aut, cre]
Maintainer Waldir Leoncio <w.l.netto@medisin.uio.no>
Repository CRAN
Date/Publication 2022-05-13 06:40:02 UTC

R topics documented:

blanks .......................................................... 2
cell ............................................................ 3
colon ........................................................ 3
find .......................................................... 4
fix ............................................................ 5
gammaln ..................................................... 5
inputdlg ..................................................... 6
isempty ...................................................... 7
issfield ...................................................... 7
isspace ...................................................... 8
blanks

Description
Create character vector of blanks

Usage
blanks(n)

Arguments
n       length of vector

Details
This function emulates the behavior of a homonymous function from Matlab

Value
Vector of n blanks

Author(s)
Waldir Leoncio
**cell**

**Examples**

blanks(1)
blanks(3)

---

**Description**

Creates an array of zeros

**Usage**

cell(n, sz = c(n, n), expandable = FALSE, ...)

**Arguments**

- `n` a the first dimension (or both, if sz is not passed)
- `sz` the second dimension (or 1st and 2nd, if not passed)
- `expandable` if TRUE, output is a list (so it can take different lengths)
- ... Other dimensions

**Value**

An array of zeroes with the dimensions passed on call

**Examples**

cell(5)
cell(5, 2)

---

**colon**

**Vector creation**

**Description**

Simulates the function `colon()` and its equivalent : operator from Matlab, which have a similar but not quite equivalent behavior when compared to `seq()` and : in R.

**Usage**

colon(a, b)
Arguments

a  initial number
b  final number

Value

A vector containing a sequence of integers going from a to b

Examples

colon(1, 4)
colon(4, 8)

find

Find indices and values of nonzero elements

Description

Emulates behavior of find

Usage

find(x, sort = TRUE)

Arguments

x  object or logic operation on an object
sort  sort output?

Value

A vector of indices of x that satisfy the logical test (nonzero, by default).

Examples

X <- matrix(c(1, 0, 2, 0, 1, 1, 0, 0, 4), 3, byrow = TRUE)
Y <- seq(1, 19, 2)
find(X)
find(Y == 13)
**fix**  
*Round toward zero*

**Description**
Rounds each element of input to the nearest integer towards zero. Basically the same as trunc()

**Usage**
fix(X)

**Arguments**

**Input element**

**Value**
The values of `trunc(X)`.

**Author(s)**
Waldir Leoncio

**Examples**

```r
X <- matrix(c(-1.9, -3.4, 1.6, 2.5, -4.5, 4.5), 3, byrow = TRUE)
Y <- matrix(c(-1, -3, 1, 2, -4, 4), 3, byrow = TRUE)
fix(X)
fix(Y)
```

---

**gammaln**  
*Logarithm of gamma function*

**Description**
Calculates the natural logarithm of the gamma function

**Usage**
gammaln(A)

**Arguments**

**A**  
a non-negative, real matrix, vector or scalar
Value

An element-by-element \( \ln(\text{gamma}()) \)-transformed A

Note

For MATLAB output reproduction, non-positive values will be

Author(s)

Waldir Leoncio

Examples

\begin{verbatim}
gammaln(8)  \hspace{1cm} \text{gamma}(8)
gammaln(0)  \hspace{1cm} \text{gamma}(0)
gammaln(matrix(1:9, 3)) \hspace{1cm} \text{gamma(matrix}(1:9, 3))
gammaln(-4:10) \hspace{1cm} \text{gamma}(-4:10)
\end{verbatim}

Description

Replicates the functionality of the homonymous function in Matlab (sans dialog box)

Usage

\begin{verbatim}
inputdlg(prompt, dims = 1, definput = NULL)
\end{verbatim}

Arguments

\begin{verbatim}
prompt \hspace{1cm} \text{Text field with user instructions}
dims \hspace{1cm} \text{number of dimensions in the answers}
definput \hspace{1cm} \text{default value of the input}
\end{verbatim}

Examples

\begin{verbatim}
## Not run:
name <- inputdlg("Type your name")
paste("Hello,", name)
## End(Not run)
\end{verbatim}
isemty

Is Array Empty?

Description

Determine whether array is empty. An empty array, table, or timetable has at least one dimension with length 0, such as 0-by-0 or 0-by-5.

Usage

isemty(x)

Arguments

x array

Details

Emulates the behavior of the isemty function on Matlab

Value

A logical value determining if x is empty

Examples

isemty(array(dim = c(0, 2, 2)))
isemty(matrix(rep(NA, 4), 2))
isemty(matrix(rep(0, 4), 2))
isemty(as.factor(c(NA, NA)))
isemty(factor())
isemty(matrix(rep(\"\", 3)))

isfield

Checks if a list contains a field

Description

This function tries to replicate the behavior of the isfield function in Matlab

Usage

isfield(x, field)
Arguments

\[ x \quad \text{list} \]
\[ \text{field} \quad \text{name of field} \]

Value

A logical vector determining if \text{field} is within \text{names}(x)

References

https://se.mathworks.com/help/matlab/ref/isfield.html

Examples

\[ S \leftarrow \text{list}( \]
\[ \quad x = \text{rnorm}(100), \]
\[ \quad \text{title} = \text{"x"} \]
\[ \) \]
\[ \text{isfield}(S, \text{"title")} \]
\[ \text{isfield}(S, \text{"z")} \]

isspace

Determine space characters

Description

Determine which characters are space characters

Usage

isspace(A)

Arguments

A \quad \text{a character array or a string scalar}

Value

a vector TF such that the elements of TF are logical 1 (true) where corresponding characters in A are space characters, and logical 0 (false) elsewhere.

Note

Recognized whitespace characters are \text{and} \text{\ and} \text{\t}.

Author(s)

Waldir Leoncio
Examples

chr <- "123 Main St."
X <- "\t a b\tcde f"
isspace(chr)
isspace(X)

Description

Performs basic syntax conversion from a Matlab function file to R

Usage

matlab2r(
  filename,
  output = "diff",
  improve_formatting = TRUE,
  change_assignment = TRUE,
  append = FALSE,
  restyle = !improve_formatting
)

Arguments

filename name of the file
output can be "asis", "clean", "save" or "diff"
improve_formatting if TRUE (default), makes minor changes to conform to best-practice formatting conventions
change_assignment if TRUE (default), uses <- as the assignment operator
append if FALSE (default), overwrites file; otherwise, append output to input
restyle if TRUE, will restyle the output with styler (only for output = "save")

Value
text converted to R, printed to screen or replacing input file

Note

This function is intended to expedite the process of converting a Matlab function to R by making common replacements. It does not have the immediate goal of outputting a ready-to-use function. In other words, after using this function you should go back to it and make minor changes.

It is also advised to do a dry-run with output = "clean" and only switching to output = "save" when you are confident that no important code will be lost (for shorter functions, a careful visual inspection should suffice).
**Author(s)**

Waldir Leoncio

**Examples**

```r
code <- system.file("extdata", "matlabDemo.m", package = "matlab2r")
m2r(code)
m2r(code, output = "clean")
```

**Description**

Finds the minimum value for each column of a matrix, potentially returning the indices instead.

**Usage**

```r
max(X, indices = TRUE)
```

**Arguments**

- `X`: matrix
- `indices`: return indices?

**Value**

Either a list or a vector

**Author(s)**

Waldir Leoncio

**Examples**

```r
A <- matrix(c(23, 42, 37, 15, 52))
max(A)
base::max(A) # for comparison
```
Description

Finds the minimum value for each column of a matrix, potentially returning the indices instead

Usage

\[
\text{min}(X, \text{indices} = \text{TRUE})
\]

Arguments

\[
\begin{align*}
X & \quad \text{matrix} \\
\text{indices} & \quad \text{return indices?}
\end{align*}
\]

Value

Either a list or a vector

Author(s)

Waldir Leoncio

Examples

\[
A \leftarrow \text{matrix}(c(23, 42, 37, 15, 52))
\]
\[
\text{min}(A)
\]
\[
\text{base::min}(A) \quad \text{# for comparison}
\]

Description

Returns the number of arguments passed to the parent function

Usage

\[
\text{nargin}()
\]

Value

An integer indicating how many input arguments a function received.
Note

This function only makes sense inside another function

Author(s)

Waldir Leoncio

References

https://stackoverflow.com/q/64422780/1169233

Examples

f <- function(x, y, z) return(nargin())
f(pi)
f(y = 6, z = 5)
f(letters)
f(letters, LETTERS, pi)

ones Matrix of ones

Description

wrapper of zeros_or_ones() that replicates the behavior of the ones() function on Matlab

Usage

ones(n1, n2 = n1, ...)

Arguments

n1 number of rows
n2 number of columns
... extra dimensions

Value

An n1-by-n2 matrix of ones

Examples

ones(3)
ones(8, 1)
### Description

This function aims to loosely mimic the behavior of the questdlg function on Matlab.

### Usage

```r
questdlg(
  quest,
  dlgtitle = "",
  btn = c("y", "n"),
  defbtn = "n",
  accepted_ans = c("y", "yes", "n", "no")
)
```

### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>quest</td>
<td>Question</td>
</tr>
<tr>
<td>dlgtitle</td>
<td>Title of question</td>
</tr>
<tr>
<td>btn</td>
<td>Vector of alternatives</td>
</tr>
<tr>
<td>defbtn</td>
<td>Scalar with the name of the default option</td>
</tr>
<tr>
<td>accepted_ans</td>
<td>Vector containing accepted answers</td>
</tr>
</tbody>
</table>

### Value

Whatever is entered by the user after the prompt created by the function.

### Examples

```r
## Not run:
ans <- questdlg("Do you want to continue?", "Continue?")
if (tolower(substring(ans, 1, 1)) == "y") {
  message("You typed yes")
} else {
  message("You didn't type yes")
}

## End(Not run)
```
repmat

**rand**

*Generate matrix with U(0, 1) trials*

**Description**

Imitates the behavior of `rand()` on Matlab

**Usage**

`rand(r = 1, c = 1)`

**Arguments**

- \( r \) number of rows of output matrix
- \( c \) number of columns of output matrix

**Value**

\( r \times c \) matrix with random trials from a standard uniform distribution.

**Examples**

- `rand()`
- `rand(3, 2)`

---

repmat

**repmat**

*Repeat matrix*

**Description**

Repeats a matrix over \( n \) columns and rows

**Usage**

`repmat(mx, n)`

**Arguments**

- \( mx \) matrix
- \( n \) either a scalar with the number of replications in both rows and columns or a \( <= 3 \)-length vector with individual repetitions.

**Details**

This function was created to replicate the behavior of a homonymous function on Matlab
Value

matrix replicated over ncol(mx) * n columns and nrow(mx) * n rows

Note

The Matlab implementation of this function accepts n with length > 2.

It should also be noted that a concatenated vector in R, e.g. c(5, 2), becomes a column vector when coerced to matrix, even though it may look like a row vector at first glance. This is important to keep in mind when considering the expected output of this function. Vectors in R make sense to be seen as column vectors, given R’s Statistics-oriented paradigm where variables are usually disposed as columns in a dataset.

Examples

```r
x <- matrix(1:4, 2)
repmat(x, 1)
repmat(x, 2)
repmat(x, c(2, 3))
```

Description

Reshapes a matrix according to a certain number of dimensions

Usage

```r
reshape(A, sz)
```

Arguments

- `A`: input matrix
- `sz`: vector containing the dimensions of the output vector

Details

This function replicates the functionality of the `reshape()` function on Matlab. This function is basically a fancy wrapper for the `array()` function in R, but is useful because it saves the user translation time. Moreover, it introduces validation code that alter the behavior of `array()` and makes it more similar to `replicate()`.

Value

the input matrix, reshaped according to the vector `sz`
Note

The Matlab function also accepts as input the dismemberment of sz as scalars.

Examples

```r
mx <- matrix(1:4, 2)
ra <- array(1:12, c(2, 3, 2))

mx
reshape(mx, c(1, 4))

ra
reshape(ra, c(3, 2, 2))
```

---

setdiff

Set differences of two arrays

Description

Loosely replicates the behavior of the homonym Matlab function

Usage

```r
setdiff(A, B, legacy = FALSE)
```

Arguments

- **A** first array
- **B** second array
- **legacy** if TRUE, preserves the behavior of the setdiff function from MATLAB R2012b and prior releases. (currently not supported)

Value

An array containing he elements which are in A but not in B

Author(s)

Waldir Leoncio

Examples

```r
A <- c(3, 6, 2, 1, 5, 1, 1)
B <- c(2, 4, 6)
setdiff(A, B)
```
**size**  

**Size of an object**

**Description**

This function tries to replicate the behavior of the base function "size" in Matlab.

**Usage**

```
size(x, d)
```

**Arguments**

- `x`: object to be evaluated
- `d`: dimension of object to be evaluated

**Value**

A vector whose size is the number of dimensions of `x` and whose scale corresponds to the number of elements on (i.e. the size of) each dimension.

**Note**

On MATLAB, `size(1, 100)` returns 1. As a matter of fact, if the user calls for a dimension which `x` doesn't have `size()` always returns 1. R’s default behavior is more reasonable in those cases (i.e., returning NA), but since the point of this function is to replicate MATLAB behaviors (bugs and questionable behaviors included), this function also does this.

**Examples**

```
size(10)
size(1:4)
size(matrix(1:6, 2))
size(array(1:24, c(2, 3, 4)))
```

---

**sortrows**  

**Sort rows of matrix or table**

**Description**

Emulates the behavior of the `sortrows` function on Matlab.

**Usage**

```
sortrows(A, column = 1)
```
Arguments

- **A**  
  matrix

- **column**  
  ordering column

Value

The A matrix sorted by the first row, then the second

Examples

```r
mx <- matrix(c(3, 2, 2, 1, 1, 10, 0, pi), 4)
mx
sortrows(mx)
```

---

**squeeze**  
*Squeeze*

Description

Remove dimensions of length 1

Usage

`squeeze(A)`

Arguments

- **A**  
  input or array matrix

Details

This function implements the behavior of the homonimous function on Matlab. $B = squeeze(A)$ returns an array with the same elements as the input array A, but with dimensions of length 1 removed. For example, if A is a 3-by-1-by-1-by-2 array, then squeeze(A) returns a 3-by-2 matrix. If A is a row vector, column vector, scalar, or an array with no dimensions of length 1, then squeeze returns the input A.

Value

An array with the same elements as the input array, but with dimensions of length 1 removed.

Note

This is basically a wrapper of drop() with a minor adjustment to adapt the output to what happens on Matlab.
**strcmp**

*Compare two character elements*

---

**Description**

Logical test if two character elements are identical

**Usage**

```
strcmp(s1, s2)
```

**Arguments**

- `s1` first character element (string, vector or matrix)
- `s2` second character element (string, vector or matrix)

**Value**

a logical element of the same type as the input

**Examples**

```
strcmp("yes", "no")
strcmp("yes", "yes")
strcmp("no", "no")
```
times  Element-wise matrix multiplication

Description

Emulates the times() and .* operators from Matlab.

Usage

times(a, b)

Arguments

a  first factor of the multiplication
b  second factor of the multiplication

Details

This function basically handles elements of different length better than the * operator in R, at least as far as behavior from a Matlab user is expecting.

Value

matrix with dimensions equal to the larger of the two factors

Examples

times(9, 6)
x <- matrix(1:4, 2)
y <- c(10, 3)
print(x)
print(y)
times(x, y)
x * y

uigetfile  Select a file for loading

Description

Loosely mimics the functionality of the uigetfile function on Matlab.

Usage

uigetfile(filter = "", title = "")
Arguments

filter Filter listed files
title Pre-prompt message

Value

A list containing the name of the file selected and its path

References

https://se.mathworks.com/help/matlab/ref/uigetfile.html

Examples

```r
## Not run:
uigetfile()
## End(Not run)
```

uiputfile

Description

This function intends to loosely mimic the behaviour of the homonymous Matlab function.

Usage

```r
uiputfile(filter = ".rda", title = "Save file")
```

Arguments

filter accepted file extension
title Title

Value

A list containing the name and the path of the file to be saved

Examples

```r
## Not run:
uigetfile()
## End(Not run)
```
zeros

*Matrix of zeros*

**Description**

Wrapper of `zeros_or_ones()` that replicates the behavior of the `zeros()` function on Matlab.

**Usage**

```
zeros(n1, n2 = n1, ...)
```

**Arguments**

- `n1`: number of rows
- `n2`: number of columns
- `...`: extra dimensions

**Value**

An `n1`-by-`n2` matrix of zeros

**Examples**

```
zeros(5)
zeros(5, 3)
```

zeros_or_ones

*Matrix of zeros or ones*

**Description**

Generates a square or rectangular matrix of zeros or ones.

**Usage**

```
zeros_or_ones(n, x)
```

**Arguments**

- `n`: scalar or 2D vector
- `x`: value to fill matrix with

**Details**

This is a wrapper function to replicate the behavior of the `zeros()` and the `ones()` functions on Matlab.
Value

n-by-n matrix filled with x

Note

Actually works for any x, but there’s no need to bother imposing validation controls here.
Index

blanks, 2

cell, 3
colon, 3

find, 4
fix, 5
gammaln, 5

inputdlg, 6
isempty, 7
isfield, 7
isspace, 8

matlab2r, 9
max, 10
min, 11

nargin, 11
ones, 12

questdlg, 13

rand, 14
repmat, 14
reshape, 15

setdiff, 16
size, 17
sortrows, 17
squeeze, 18
strcmp, 19

times, 20

uigetfile, 20
uiputfile, 21

zeros, 22
zeros_or_ones, 22