

# Package ‘matlab2r’

January 31, 2023

**Title** Translation Layer from MATLAB to R

**Version** 1.5.0

**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** 7.2.3

**Suggests** testthat (>= 3.0.0)

**Config/testthat/edition** 3

**Imports** methods, styler

**URL** <https://ocbe-uio.github.io/matlab2r/>

**BugReports** <https://github.com/ocbe-uio/matlab2r/issues>

**NeedsCompilation** no

**Author** Waldir Leoncio [aut, cre]

**Maintainer** Waldir Leoncio <w.l.netto@medisin.uio.no>

**Repository** CRAN

**Date/Publication** 2023-01-31 14:10:02 UTC

## R topics documented:

.onAttach	2
assert	3
blanks	4
cell	4
char	5
colon	6
disp	6
find	7

fix . . . . .	8
gammaln . . . . .	8
inputdlg . . . . .	9
isempty . . . . .	10
isfield . . . . .	10
isFilePath . . . . .	11
ismember . . . . .	12
ismembertol . . . . .	13
isspace . . . . .	14
linspace . . . . .	14
log2 . . . . .	15
matlab2r . . . . .	16
max . . . . .	17
min . . . . .	18
nargin . . . . .	18
num2str . . . . .	19
ones . . . . .	20
questdlg . . . . .	21
rand . . . . .	22
rem . . . . .	23
repmat . . . . .	23
reshape . . . . .	24
setdiff . . . . .	25
size . . . . .	26
sortrows . . . . .	27
squeeze . . . . .	27
strcmp . . . . .	28
sum_MATLAB . . . . .	29
times . . . . .	30
uigetfile . . . . .	30
uiputfile . . . . .	31
zeros . . . . .	32
zeros_or_ones . . . . .	32

**Index** **34**

---

.onAttach	<i>Prints welcome message on package load</i>
-----------	---

---

**Description**

Prints package version number and welcome message on package load

**Usage**

.onAttach(libname, pkgname)

**Arguments**

libname	library location. See ?base::.onAttach for details
pkgname	package name. See ?base::.onAttach for details

---

assert	<i>Assert if condition is true</i>
--------	------------------------------------

---

**Description**

Throw error if condition false

**Usage**

```
assert(cond, msg = "Assertion failed.", A = NULL)
```

**Arguments**

cond	Logical test
msg	Error message to be displayed if cond == FALSE
A	values to format msg if the latter contains C-style formatting commands. formatted as parsable

**Value**

The error message if cond == FALSE, nothing otherwise

**Author(s)**

Waldir Leoncio

**Examples**

```
minVal <- 7
x <- 26
assert(minVal < x) # should return nothing
maxVal <- 13
## Not run:
  assert((minVal < x) && (x < maxVal))
  assert(x == "a", "x is %s", class(x))

## End(Not run)
```

---

blanks	<i>Blanks</i>
--------	---------------

---

**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

**Examples**

```
blanks(1)
blanks(3)
```

---

cell	<i>Cell array</i>
------	-------------------

---

**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)
```

---

char *Convert an array to a character array*

---

**Description**

A character array is a sequence of characters, just as a numeric array is a sequence of numbers. A typical use is to store a short piece of text as a row of characters in a character vector.

**Usage**

```
char(A)

## S4 method for signature 'character'
char(A)

## S4 method for signature 'array'
char(A)
```

**Arguments**

A a vector or array (not yet supported)

**Value**

A converted to characters

**Methods (by class)**

- char(character): Converting a character vector
- char(array): Converting a character array

**Author(s)**

Waldir Leoncio

**Examples**

```
char("Hi!")  
char(matrix(letters, 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)
```

---

disp

*Display the value of a variable*

---

**Description**

`disp(X)` displays the value of variable `X` without printing the variable name. This is a wrapper around base `::cat()` that includes a breakline in the end.

**Usage**

```
disp(X)
```

**Arguments**

X                    variable

**Value**

The value of X

**Author(s)**

Waldir Leoncio

**Examples**

```
A <- c(15, 150)
S <- 'Hello World.'
disp(A)
disp(S)
```

---

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**

X                   input element

**Value**

The values of trunc(X).

**Author(s)**

Waldir Leoncio

**Examples**

```
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**

```
gammaIn(8)
gammaIn(0)
gammaIn(matrix(1:9, 3))
gammaIn(-4:10)
```

---

inputdlg

*Gather user input*

---

**Description**

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

**Usage**

```
inputdlg(prompt, dims = 1, definput = NULL)
```

**Arguments**

prompt	Text field with user instructions
dims	number of dimensions in the answers
definput	default value of the input

**Value**

A user prompt

**Examples**

```
## Not run:
name <- inputdlg("Type your name")
paste("Hello,", name)

## End(Not run)
```

---

`isempty`*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**`isempty(x)`**Arguments**`x`                    array**Details**

Emulates the behavior of the `isempty` function on Matlab

**Value**

A logical value determining if `x` is empty

**Examples**

```
isempty(array(dim = c(0, 2, 2)))
isempty(matrix(rep(NA, 4), 2))
isempty(matrix(rep(0, 4), 2))
isempty(as.factor(c(NA, NA)))
isempty(factor())
isempty(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	list
field	name of field

**Value**

A logical vector determining if field is within names(x)

**References**

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

**Examples**

```
S <- list(
  x = rnorm(100),
  title = "x"
)
isfield(S, "title")
isfield(S, "z")
```

---

isFilePath

*Check if an input is a valid path*

---

**Description**

A simple check if an input corresponds to a valid path to a file

**Usage**

```
isFilePath(x)
```

**Arguments**

x	input
---	-------

**Value**

TRUE if x is a valid path, FALSE otherwise

**Author(s)**

Waldir Leoncio

---

ismember	<i>Array elements that are members of set array</i>
----------	---

---

**Description**

Checks which members of one entity are in another

**Usage**

```
ismember(A, B, rows = FALSE, indices = FALSE)
```

```
## S4 method for signature 'data.frame,data.frame'
```

```
ismember(A, B, rows = FALSE, indices = FALSE)
```

**Arguments**

A	a vector, matrix or dataframe
B	another vector, matrix or dataframe
rows	if TRUE, each row of A and each row of B are treated as single entities
indices	if TRUE, outputs the lowest B index for each match in A

**Value**

a binary vector telling if the corresponding A indices are in B. If indices = TRUE, also prints the index in B where the match first occurs.

**Author(s)**

Waldir Leoncio

**Examples**

```
# Values that are members of set
A <- c(5, 3, 4, 2)
B <- c(2, 4, 4, 4, 6, 8)
ismember(A, B)

# Members of set and indices to values
ismember(A, B, indices = TRUE)

# Table rows found in another table
A <- data.frame(
  "V1" = 1:5, "V2" = LETTERS[1:5], "V3" = as.logical(c(0, 1, 0, 1, 0))
)
B <- data.frame(
  "V1" = seq(1, 9, 2), "V2" = LETTERS[seq(1, 9, 2)], "V3" = as.logical(rep(0, 5))
)
ismember(A, B)
```

---

ismembertol	<i>Tolerant alternative to ismember</i>
-------------	---

---

**Description**

Does the

**Usage**

```
ismembertol(A, B, rows = FALSE, indices = TRUE)
```

**Arguments**

A	a vector, matrix or dataframe
B	another vector, matrix or dataframe
rows	if TRUE, each row of A and each row of B are treated as single entities
indices	if TRUE, outputs the lowest B index for each match in A

**Value**

Same as ismember

**Author(s)**

Waldir Leoncio

**See Also**

ismember

**Examples**

```
x <- t(1:6) * pi
y <- 10 ^ log10(x)

# Show that values are equal, but not identical (due to floating-point error)
all.equal(x, y)
identical(x, y)

# Checking the difference in outputs
ismember(x, y)
ismembertol(x, y)
```

isspace

*Determine space characters*

---

**Description**

Determine which characters are space characters

**Usage**

```
isspace(A)
```

**Arguments**

A                    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  and `\\t`.

**Author(s)**

Waldir Leoncio

**Examples**

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

---

linspace

*Generate linearly-spaced vector*

---

**Description**

This is a soft wrap around the `base::seq()` function.

**Usage**

```
linspace(x1, x2, n = 100L)
```

**Arguments**

x1	start point
x2	end point
n	length of output

**Value**

A numeric vector of n numbers between x1 and x2.

**Author(s)**

Waldir Leoncio

**Examples**

```
linspace(-5, 4)
linspace(1 + 2i, 9 + 9i, 5)
```

---

log2

*Base 2 logarithm*


---

**Description**

Base 2 logarithm and floating-point number dissection

**Usage**

```
log2(X, dissect = TRUE)
```

**Arguments**

X	a scalar or vector of numbers
dissect	if TRUE, returns the mantissa and exponent.

**Value**

either a vector or a list of mantissas and exponents such that mantissa \* 2 ^ exponent equals X

**Examples**

```
log2(10, dissect = FALSE)
log2(10)
.625 * 2 ^ 4 == 10 # proof
```

matlab2r

*Convert Matlab function to R***Description**

Performs basic syntax conversion from a Matlab function file to R

**Usage**

```
matlab2r(
  input,
  output = "diff",
  improve_formatting = TRUE,
  change_assignment = TRUE,
  append = FALSE,
  restyle = !improve_formatting,
  skip_lines = NULL
)
```

**Arguments**

input	file path or character string containing MATLAB code
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")
skip_lines	vector of lines to be skipped. These will be commented out and tagged as TODO, instead.

**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**

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

---

max

*Maximum (MATLAB version)*

---

**Description**

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

**Usage**

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

**Arguments**

X	matrix
indices	return indices?

**Value**

Either a list or a vector

**Author(s)**

Waldir Leoncio

**Examples**

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

min *Minimum (MATLAB version)*

---

**Description**

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

**Usage**

```
min(X, indices = TRUE)
```

**Arguments**

X	matrix
indices	return indices?

**Value**

Either a list or a vector

**Author(s)**

Waldir Leoncio

**Examples**

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

---

nargin *Number of function input arguments*

---

**Description**

Returns the number of arguments passed to the parent function

**Usage**

```
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)
```

---

num2str

*Numeric to string*

---

**Description**

Converts a numeric value to character. This is essentially a wrapper over `base::as.character()`.

**Usage**

```
num2str(A, format)

## S4 method for signature 'numeric,missing'
num2str(A)

## S4 method for signature 'array,missing'
num2str(A)

## S4 method for signature 'numeric,numeric'
num2str(A, format)

## S4 method for signature 'array,numeric'
num2str(A, format)

## S4 method for signature 'numeric,character'
num2str(A, format)

## S4 method for signature 'array,character'
num2str(A, format)
```

**Arguments**

A                    numeric object  
format               either a number or a string (see `fmt` argument of `base::sprintf()`).

**Value**

A, with its format possibly reshaped by `format`

**Methods (by class)**

- `num2str(A = numeric, format = missing)`: Converting a vector to character
- `num2str(A = array, format = missing)`: Converting an array to character
- `num2str(A = numeric, format = numeric)`: Rounding a vector, then converting to character
- `num2str(A = array, format = numeric)`: Rounding an array, then converting to character
- `num2str(A = numeric, format = character)`: Formatting a vector, then converting to character
- `num2str(A = array, format = character)`: Formatting an array, then converting to character

**Author(s)**

Waldir Leoncio

**Examples**

```
X <- rnorm(10)
num2str(X)
num2str(X, 2)
A <- matrix(runif(4), 2)
num2str(A)
num2str(A, 3)
num2str(pi * 10, "%e")
```

---

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)
```

---

questdlg	<i>Prompt for multiple-choice</i>
----------	-----------------------------------

---

**Description**

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

**Usage**

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

**Arguments**

quest	Question
dlgtitle	Title of question
btn	Vector of alternatives
defbtn	Scalar with the name of the default option
accepted_ans	Vector containing accepted answers

**Value**

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

## Examples

```
## 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)
```

---

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)
```

---

rem	<i>Remainder after division</i>
-----	---------------------------------

---

**Description**

Returns the remainder after division of a by b, where a is the dividend and b is the divisor. This function is often called the remainder operation. The rem function follows the convention that `rem(a, 0)` is NaN.

**Usage**

```
rem(a, b)
```

**Arguments**

a	the dividend
b	the divisor

**Value**

The remainder

**Author(s)**

Waldir Leoncio

**Examples**

```
rem(23, 5)
rem(1:5, 3)
rem(c(-4, -1, 7, 9), 3) #FIXME
rem(c(0, 3.5, 5.9, 6.2, 9, 4 * pi), 2 * pi)
```

---

repmat	<i>Repeat matrix</i>
--------	----------------------

---

**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 $\leq 3$ -length vector with individual repetitions.

**Details**

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

**Value**

matrix replicated over  $\text{ncol}(mx) * n$  columns and  $\text{nrow}(mx) * n$  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**

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

---

reshape

*Reshape array*

---

**Description**

Reshapes a matrix according to a certain number of dimensions

**Usage**

```
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**

```
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**

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

**Arguments**

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

**Value**

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

**Author(s)**

Waldir Leoncio

**Examples**

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

---

**size***Size of an object*

---

**Description**

This functions 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	<i>Sort rows of matrix or table</i>
----------	-------------------------------------

---

**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**

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

---

squeeze	<i>Squeeze</i>
---------	----------------

---

**Description**

Remove dimensions of length 1

**Usage**

```
squeeze(A)
```

**Arguments**

A	input or array matrix
---	-----------------------

**Details**

This function implements the behavior of the homonymous 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

**Author(s)**

Waldir Leoncio

**Examples**

```
A <- array(dim = c(2, 1, 2))
A[, , 1] <- c(1, 2)
A[, , 2] <- c(3, 4)
print(A)
squeeze(A)
```

---

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")
```

---

sum_MATLAB	<i>Sum of array elements</i>
------------	------------------------------

---

### Description

Returns the sum of the elements of the first input

### Usage

```
sum_MATLAB(A, dim)

## S4 method for signature 'array,missing'
sum_MATLAB(A)

## S4 method for signature 'array,character'
sum_MATLAB(A, dim)

## S4 method for signature 'array,numeric'
sum_MATLAB(A, dim)
```

### Arguments

A	vector, matrix or array
dim	dimension over which A is to be summed

### Value

The total, row or column sum of A

### Methods (by class)

- `sum_MATLAB(A = array, dim = missing)`: Sum elements of A along the first array dimension whose size does not equal 1
- `sum_MATLAB(A = array, dim = character)`: Computes the sum of all elements of A
- `sum_MATLAB(A = array, dim = numeric)`: Computes the sum of all elements of A

### Author(s)

Waldir Leoncio

### Examples

```
x1 <- array(1:9, c(3, 3))
sum_MATLAB(x1)
sum_MATLAB(x1, "all")
sum_MATLAB(x1, 2)
```

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**

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

---

uiputfile	<i>Save file</i>
-----------	------------------

---

**Description**

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

**Usage**

```
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**

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

---

zeros	<i>Matrix of zeros</i>
-------	------------------------

---

**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	<i>Matrix of zeros or ones</i>
---------------	--------------------------------

---

**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

.onAttach, 2  
assert, 3  
base::as.character(), 19  
base::seq(), 14  
base::sprintf(), 20  
blanks, 4  
cell, 4  
char, 5  
char, array-method (char), 5  
char, character-method (char), 5  
colon, 6  
disp, 6  
find, 7  
fix, 8  
gammaIn, 8  
inputdlg, 9  
isempty, 10  
isfield, 10  
isFilePath, 11  
ismember, 12  
ismember, data.frame, data.frame-method  
(ismember), 12  
ismembertol, 13  
isspace, 14  
linspace, 14  
log2, 15  
matlab2r, 16  
max, 17  
min, 18  
nargin, 18  
num2str, 19  
num2str, array, character-method  
(num2str), 19  
num2str, array, missing-method (num2str),  
19  
num2str, array, numeric-method (num2str),  
19  
num2str, numeric, character-method  
(num2str), 19  
num2str, numeric, missing-method  
(num2str), 19  
num2str, numeric, numeric-method  
(num2str), 19  
ones, 20  
questdlg, 21  
rand, 22  
rem, 23  
repmat, 23  
reshape, 24  
setdiff, 25  
size, 26  
sortrows, 27  
squeeze, 27  
strcmp, 28  
sum\_MATLAB, 29  
sum\_MATLAB, array, character-method  
(sum\_MATLAB), 29  
sum\_MATLAB, array, missing-method  
(sum\_MATLAB), 29  
sum\_MATLAB, array, numeric-method  
(sum\_MATLAB), 29  
times, 30  
uigetfile, 30  
uinputfile, 31  
zeros, 32  
zeros\_or\_ones, 32