

# Package ‘docore’

February 1, 2022

**Type** Package

**Title** Utility Functions for Scientific Coding

**Version** 1.0

**Author** Danail Obreschkow

**Maintainer** Danail Obreschkow <danail.obreschkow@gmail.com>

**Description** Basic routines used in scientific coding, such as timing routines, vector/array handling functions and I/O support routines.

**Imports** utils, pracma, bit64

**License** GPL-3

**Encoding** UTF-8

**RoxygenNote** 7.1.1

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2022-02-01 09:20:11 UTC

## R topics documented:

cshift . . . . .	2
last . . . . .	2
lim . . . . .	3
linuxspaces . . . . .	3
loadbin . . . . .	4
midseq . . . . .	5
quiet . . . . .	5
tick . . . . .	6
tock . . . . .	7
uniquedouble . . . . .	8
<b>Index</b>	<b>9</b>

---

cshift	<i>Circularly shift each dimension of an array</i>
--------	--

---

**Description**

Circulates each dimension of an array. This routine is identical to [circshift](#), but works with arrays up to 5 dimensions.

**Usage**

```
cshift(x, s)
```

**Arguments**

x	vector or array (up to rank 5)
s	scalar, if x is a vector, or a vector of length matching the rank of x, if x is an array

**Value**

Returns a vector or array of the same shape as x.

**Author(s)**

Danail Obreschkow

---

last	<i>Last element of a vector</i>
------	---------------------------------

---

**Description**

Returns the last element of a vector or the n-th element counting from the end of a vector.

**Usage**

```
last(x, n = 1)
```

**Arguments**

x	vector
n	optional integer specifying the n-th element from the end to be returned

**Value**

scalar of the same type as x

**Author(s)**

Danail Obreschkow

---

lim	<i>Crop values of vector or array to a custom range</i>
-----	---

---

**Description**

limits the values of a vector or array to a desired interval, while keeping the shape of the vector/array

**Usage**

```
lim(x, min = 0, max = 1, clip = NULL, na = NULL)
```

**Arguments**

x	vector or array
min	minimum value
max	maximum value
clip	optional value specifying the value assigned to clipped data, e.g. clip=NA
na	optional value specifying the value assigned to non-numbers (NA and NaN)

**Value**

vector/array of the same shape as x

**Author(s)**

Danail Obreschkow

---

linuxspaces	<i>Handle spaces in Linux filenames</i>
-------------	---

---

**Description**

Convert spaces in filenames (" ") to linux-type spaces "\ ", needed when calling system() on macOS.

**Usage**

```
linuxspaces(txt)
```

**Arguments**

txt	filename, which may contain ordinary spaces, e.g. "my file 1.txt"
-----	---

**Value**

filename with modified spaces, e.g. "my\ file\ 1.txt"

**Author(s)**

Danail Obreschkow

**Examples**

```
filename = '~/Desktop/my file 1.txt'
command = sprintf('ls -l %s',linuxspaces(filename))
## Not run:
system(command)

## End(Not run)
```

---

loadbin	<i>Read binary data into array</i>
---------	------------------------------------

---

**Description**

Reads binary data using the base function `readBin` and recasts it into an array of custom dimensions.

**Usage**

```
loadbin(
  filename,
  dim,
  bytes = 4,
  type = "numeric",
  signed = FALSE,
  endian = "little"
)
```

**Arguments**

filename	path of the file to be loaded
dim	vector specifying the dimensions of the array
bytes	number of bytes per number in the binary file
type	character vector of length describing the data type: "numeric" (default), "double", "integer", "int", "logical", "complex", "character", "raw"
signed	logical. Only used for integers of sizes 1 and 2, when it determines if the quantity on file should be regarded as a signed or unsigned integer.
endian	endian-type ("big" or "little") of the file

**Value**

Returns an array of dimension dim.

**Author(s)**

Danail Obreschkow

---

midseq	<i>Mid-points of regular grid</i>
--------	-----------------------------------

---

**Description**

compute the mid-point positions of a one-dimensional regular grid of n equal intervals.

**Usage**

```
midseq(min, max, n = 1)
```

**Arguments**

min	left boundary of first bin
max	right boundary of last bin
n	number of bins

**Value**

vector of mid points

**Author(s)**

Danail Obreschkow

---

quiet	<i>Suppress in-routine output</i>
-------	-----------------------------------

---

**Description**

Runs any routine or command while supressing in-routine console output

**Usage**

```
quiet(x)
```

**Arguments**

x	routine to be called
---	----------------------

**Value**

Returns whatever the called routine returns in invisible form.

**Author(s)**

Danail Obreschkow

**Examples**

```
# Test function
test = function(x) {
  cat('This routine is likes to talk a lot!\n')
  return(x^2)
}

# Standard call call:
y = test(5)
print(y)

# Quiet call:
y = quiet(test(6))
print(y)
```

---

tick

*Start timer*

---

**Description**

Start timer and write a custom text into the console.

**Usage**

```
tick(txt = "Start")
```

**Arguments**

txt                    custom text

**Value**

None

**Author(s)**

Danail Obreschkow

**See Also**

[tock](#)

## Examples

```
tick('Sum 10 million random numbers')
x = sum(runif(1e7))
tock()
```

---

tock

*Stop timer*

---

## Description

Stop timer and write the computation in seconds since the last call of tick().

## Usage

```
tock(txt = "")
```

## Arguments

txt                    optional custom text to be displayed

## Value

None

## Author(s)

Danail Obreschkow

## See Also

[tick](#)

## Examples

```
tick('Sum 10 million random numbers')
x = sum(runif(1e7))
tock()
```

uniquedouble

*Turn a 64-bit integer into a unique double value*

---

**Description**

Turns a 64-bit integers into unique doubles for faster comparison. The output double values are completely different from the input values.

**Usage**

```
uniquedouble(int64)
```

**Arguments**

`int64`            input value (normally used with 64-bit integers, but also works with other types)

**Value**

Returns a double floating point value.

**Author(s)**

Danail Obreschkow

**Examples**

```
# The comparison of in-built types is very fast:
int32 = as.integer(0) # (same as int32 = 0)
system.time(for(i in seq(1e4)) comparison=int32==int32)

# The comparison of 64-bit integers is very slow:
int64 = bit64::as.integer64(0)
system.time(for(i in seq(1e4)) comparison=int64==int64)

# The comparison of converted 64-bit integers is again fast:
int64d = uniquedouble(int64)
system.time(for(i in seq(1e4)) comparison=int64d==int64d)
```



# Index

circshift, 2  
cshift, 2

last, 2  
lim, 3  
linuxspaces, 3  
loadbin, 4

midseq, 5

quiet, 5

readBin, 4

tick, 6, 7  
tock, 6, 7

uniquedouble, 8