

Package ‘kimisc’

July 22, 2025

Type Package

Title Kirill's Miscellaneous Functions

Version 1.0.0

Date 2024-11-30

Description A collection of useful functions not found anywhere else, mainly for programming: Pretty intervals, generalized lagged differences, checking containment in an interval, and an alternative interface to assign().

License MIT + file LICENSE

URL <https://krlmlr.github.io/kimisc/>, <https://github.com/krlmlr/kimisc>

BugReports <https://github.com/krlmlr/kimisc/issues>

Imports memoise, plyr, pryr

Suggests testthat (>= 3.0.0)

Enhances knitr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2.9000

NeedsCompilation no

Author Kirill Müller [aut, cre]

Maintainer Kirill Müller <kirill@cynkra.com>

Repository CRAN

Date/Publication 2024-12-02 09:30:02 UTC

Contents

cut_format	2
export	3
export.list	4
gdiff	5

in.interval.lo	6
in.interval.ro	6
kimisc-deprecated	7
nin.interval.lo	7
nin.interval.ro	8

Index 9

cut_format	<i>Convert Numeric to Factor, with custom formatting</i>
------------	--

Description

This is an enhanced version of `base::cut()` that allows a custom formatting to be applied to the values.

Usage

```
cut_format(
  x,
  breaks,
  include.lowest = FALSE,
  right = TRUE,
  ordered_result = FALSE,
  ...,
  format_fun = format,
  sep = ", ",
  paren = c("(", "[", ")", "]")
)
```

Arguments

<code>x</code>	a numeric vector which is to be converted to a factor by cutting.
<code>breaks</code>	[numeric] A vector of two or more unique cut points
<code>include.lowest</code>	logical, indicating if an 'x[i]' equal to the lowest (or highest, for <code>right = FALSE</code>) 'breaks' value should be included.
<code>right</code>	logical, indicating if the intervals should be closed on the right (and open on the left) or vice versa.
<code>ordered_result</code>	logical: should the result be an ordered factor?
<code>...</code>	Passed to <code>cut()</code>
<code>format_fun</code>	[function(x): character] A vectorized function that performs the desired formatting. Default: <code>base::format()</code>
<code>sep</code>	[character(1)] The separator between lower and upper end of the interval. Default: <code>" , "</code>
<code>paren</code>	[character(4)] Opening and closing parentheses in two variants. Default: <code>c("(", "[", ")", "]")</code>

See Also

<https://stackoverflow.com/q/14456371/946850>

Examples

```
cut_format(runif(10), seq(0, 1, by = 0.25), format_fun = function(x) paste(x * 100, "%"))
cut_format(runif(10), seq(0, 1, by = 0.25), paren = c("<", "{", ">", "}")
```

export

Exports to an environment

Description

This function is a wrapper around `export.list()` that exports variables by their name to another environment.

Usage

```
export(..., target.env = .GlobalEnv)
```

Arguments

... variables to be exported.
target.env The target environment. Use the global environment by default.

Value

Invisible NULL.

Author(s)

Roland

References

<https://stackoverflow.com/a/17484932/946850>

See Also

`export.list()`, `assign()`

Examples

```
local({
  newly.created.var <- 5
  export(newly.created.var)
})
newly.created.var
rm(newly.created.var)
```

export.list	<i>Exports to an environment</i>
-------------	----------------------------------

Description

This function is a wrapper around `assign()` that exports the contents of a named list to an environment. The variable names in the target environment are constructed from the names of the list items or taken from a separate argument.

Usage

```
export.list(arg.list, arg.names = names(arg.list), target.env = .GlobalEnv)
```

Arguments

<code>arg.list</code>	list of objects, possibly named.
<code>arg.names</code>	names to use for the items in the target environment. Use the names of <code>arg.list</code> by default.
<code>target.env</code>	The target environment. Use the global environment by default.

Value

Invisible NULL.

Author(s)

Roland

References

<https://stackoverflow.com/a/17484932/946850>

See Also

`export()`, `assign()`

Examples

```
export.list(list(newly.created.var = 5))
newly.created.var
rm(newly.created.var)
```

gdiff	<i>Generalized lagged differences</i>
-------	---------------------------------------

Description

Returns suitably lagged and iterated differences using arbitrary difference functions.

Usage

```
gdiff(x, lag = 1L, differences = 1L, FUN = `~`, ...)
```

Arguments

x	a numeric vector or matrix containing the values to be differenced.
lag	an integer indicating which lag to use.
differences	an integer indicating the order of the difference.
FUN	A distance function that accepts two parameters
...	further arguments to be passed to or from methods.

Value

If x is a vector of length n and differences = 1, then the computed result is equal to the successive differences FUN(x[(1+lag):n], x[1:(n-lag)]).

If difference is larger than one this algorithm is applied recursively to x. Note that the returned value is a vector which is shorter than x.

If x is a matrix then the difference operations are carried out on each column separately.

See Also

[base::diff\(\)](#)

Examples

```
gdiff(1:4)
gdiff(1:4, FUN = `~/`)
```

<code>in.interval.lo</code>	<i>Checks if values are contained in an interval (open on the left)</i>
-----------------------------	---

Description

This function checks if the values in the `x` parameter are contained in the interval $(lo, hi]$. NA values are treated as "not in the interval".

Usage

```
in.interval.lo(x, lo, hi)
```

Arguments

<code>x</code>	A vector of values. (Lists will be coerced to a numeric vector.)
<code>lo</code>	Left end of the interval.
<code>hi</code>	Right end of the interval.

Value

A boolean vector of the same length as `x`.

See Also

[in.interval.ro\(\)](#), [nin.interval.lo\(\)](#), [nin.interval.ro\(\)](#)

Examples

```
in.interval.lo(c(-1, 0, 1, 2), 0, 1)
in.interval.lo(NA, 1, 3)
```

<code>in.interval.ro</code>	<i>Checks if values are contained in an interval (open on the right)</i>
-----------------------------	--

Description

This function checks if the values in the `x` parameter are contained in the interval $[lo, hi)$. NA values are treated as "not in the interval".

Usage

```
in.interval.ro(x, lo, hi)
```

Arguments

x	A vector of values. (Lists will be coerced to a numeric vector.)
lo	Left end of the interval.
hi	Right end of the interval.

Value

A boolean vector of the same length as x.

See Also

[in.interval.lo\(\)](#), [nin.interval.lo\(\)](#), [nin.interval.ro\(\)](#)

Examples

```
in.interval.ro(c(-1, 0, 1, 2), 0, 1)
in.interval.ro(NA, 1, 3)
```

kimisc-deprecated *Deprecated functions*

Description

The "See also" section contains the deprecated functions in this package.

See Also

Other deprecated functions: [coalesce.na-deprecated](#), [df_to_list-deprecated](#), [hms.to.seconds-deprecated](#), [list_to_df-deprecated](#), [nc-deprecated](#), [nlist-deprecated](#), [ofactor-deprecated](#), [sample.rows-deprecated](#), [seconds.to.hms-deprecated](#), [thisfile-deprecated](#), [tll-deprecated](#), [vswitch-deprecated](#)

`nin.interval.lo` *Checks if values are outside of an interval (open on the left)*

Description

This function checks if the values in the x parameter are contained in the interval (lo, hi]. NA values are treated as "not in the interval".

Usage

```
nin.interval.lo(x, lo, hi)
```

Arguments

x	A vector of values. (Lists will be coerced to a numeric vector.)
lo	Left end of the interval.
hi	Right end of the interval.

Value

A boolean vector of the same length as x.

See Also

[in.interval.lo\(\)](#), [in.interval.ro\(\)](#), [nin.interval.ro\(\)](#)

Examples

```
nin.interval.lo(c(-1, 0, 1, 2), 0, 1)
nin.interval.lo(NA, 1, 3)
```

nin.interval.ro	<i>Checks if values are outside of an interval (open on the right)</i>
-----------------	--

Description

This function checks if the values in the x parameter are contained in the interval [lo, hi). NA values are treated as "not in the interval".

Usage

```
nin.interval.ro(x, lo, hi)
```

Arguments

x	A vector of values. (Lists will be coerced to a numeric vector.)
lo	Left end of the interval.
hi	Right end of the interval.

Value

A boolean vector of the same length as x.

See Also

[in.interval.lo\(\)](#), [in.interval.ro\(\)](#), [nin.interval.lo\(\)](#)

Examples

```
nin.interval.ro(c(-1, 0, 1, 2), 0, 1)
nin.interval.ro(NA, 1, 3)
```


Index

* deprecated functions

kimisc-deprecated, 7

assign(), 3, 4

base::cut(), 2

base::diff(), 5

base::format(), 2

cut_format, 2

export, 3

export(), 4

export.list, 4

export.list(), 3

gdiff, 5

in.interval.lo, 6

in.interval.lo(), 7, 8

in.interval.ro, 6

in.interval.ro(), 6, 8

kimisc-deprecated, 7

nin.interval.lo, 7

nin.interval.lo(), 6–8

nin.interval.ro, 8

nin.interval.ro(), 6–8