Package 'kimisc'

December 2, 2024

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` *Convert Numeric to Factor, with custom formatting*

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

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

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` *Exports to an environment*

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

`arg.list`	list of objects, possibly named.
`arg.names`	names to use for the items in the target environment. Use the names of `arg.list` by default.
`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()`, `assign()`

Examples

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

gdiff *Generalized lagged differences*

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 = `/`)
```

in.interval.lo *Checks if values are contained in 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

```
in.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.ro(), nin.interval.lo(), nin.interval.ro()

Examples

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

in.interval.ro *Checks if values are contained in an interval (open on the right)*

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 *Checks if values are outside of an interval (open on the right)*

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