

Package: arkhe (via r-universe)

July 21, 2024

Title Tools for Cleaning Rectangular Data

Version 1.6.0

Maintainer Nicolas Frerebeau

<nicolas.frerebeau@u-bordeaux-montaigne.fr>

Description A dependency-free collection of simple functions for cleaning rectangular data. This package allows to detect, count and replace values or discard rows/columns using a predicate function. In addition, it provides tools to check conditions and return informative error messages.

License GPL (>= 3)

URL <https://packages.tesselle.org/arkhe/>,
<https://github.com/tesselle/arkhe>

BugReports <https://github.com/tesselle/arkhe/issues>

Depends R (>= 3.5)

Imports graphics, methods, stats, utils

Suggests tinytest

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.0

Collate 'AllGenerics.R' 'append.R' 'arkhe-deprecated.R'
'arkhe-internal.R' 'arkhe-package.R' 'predicates.R' 'assert.R'
'assign.R' 'clean.R' 'compact.R' 'conditions.R' 'count.R'
'describe.R' 'detect.R' 'discard.R' 'get.R' 'keep.R'
'mathematics.R' 'palettes.R' 'plot.R' 'remove.R' 'replace.R'
'seek.R' 'sparsity.R' 'statistics.R' 'transform.R'
'utilities.R' 'zzz.R'

Repository <https://tesselle.r-universe.dev>

RemoteUrl <https://github.com/tesselle/arkhe>

RemoteRef v1.6.0

RemoteSha fcb080b9d929d6a9668a70f1a267e6efa6bd75f6

Contents

append	3
assert_constant	4
assert_data	5
assert_length	5
assert_lower	6
assert_names	7
assert_numeric	8
assert_package	9
assert_square	10
assert_type	10
assign	11
bootstrap	12
clean_whitespace	13
compact	14
concat	16
confidence_binomial	16
confidence_mean	17
confidence_multinomial	19
count	20
describe	21
detect	22
discard	23
get	25
interval_credible	26
interval_hdr	27
is_scalar	28
jackknife	29
keep	30
math_gcd	32
math_lcm	33
null	33
palette_color	34
palette_shape	35
palette_size	35
predicate-matrix	36
predicate-numeric	37
predicate-trend	38
predicate-type	39
predicate-utils	40
remove_constant	41
remove_empty	42
remove_Inf	43
remove_NA	44
remove_zero	45
replace_empty	46
replace_Inf	47

<i>append</i>	3
replace_NA	48
replace_zero	49
scale_range	50
seek	51
sparsity	52
validate	53
Index	54

append	<i>Convert Row Names to an Explicit Column</i>
--------	--

Description

Convert Row Names to an Explicit Column

Usage

```
append_rownames(x, ...)
```

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

```
append_rownames(x, after = 0, remove = TRUE, var = "rownames")
```

Arguments

x	A data.frame .
...	Currently not used.
after	A length-one numeric vector specifying a subscript, after which the row names are to be appended.
remove	A logical scalar: should the row names be removed?
var	A character string giving the name of column to use for row names.

Value

A [data.frame](#).

Author(s)

N. Frerebeau

See Also

Other data preparation tools: [assign\(\)](#), [compact\(\)](#), [count\(\)](#), [detect\(\)](#), [discard\(\)](#), [get\(\)](#), [keep\(\)](#), [seek\(\)](#)

Examples

```
X <- data.frame(
  x = 1:5,
  y = 6:10,
  z = LETTERS[1:5]
)

## Assign column to row names
(Y <- assign_rownames(X, 3))

## Append row names to data.frame
(Z <- append_rownames(Y))
```

assert_constant	<i>Check Numeric Trend</i>
-----------------	----------------------------

Description

Check Numeric Trend

Usage

```
assert_constant(x, ...)
assert_decreasing(x, ...)
assert_increasing(x, ...)
```

Arguments

x	A numeric object to be checked.
...	Extra parameters to be passed to internal methods.

Value

Throws an error, if any, and returns x invisibly otherwise.

Author(s)

N. Frerebeau

See Also

Other validation methods: [assert_data](#), [assert_length\(\)](#), [assert_lower\(\)](#), [assert_names\(\)](#), [assert_numeric](#), [assert_package\(\)](#), [assert_square\(\)](#), [assert_type\(\)](#), [validate\(\)](#)

`assert_data`*Check Data*

Description

- `assert_missing()` and `assert_infinite()` check if an object contains any missing (NA, NaN) or infinite (Inf) value.
- `assert_unique()` checks if an object contains duplicated elements.

Usage`assert_missing(x)``assert_infinite(x)``assert_unique(x)`**Arguments**

`x` An object to be checked.

Value

Throws an error, if any, and returns `x` invisibly otherwise.

Author(s)

N. Frerebeau

See Also

Other validation methods: [assert_constant\(\)](#), [assert_length\(\)](#), [assert_lower\(\)](#), [assert_names\(\)](#), [assert_numeric](#), [assert_package\(\)](#), [assert_square\(\)](#), [assert_type\(\)](#), [validate\(\)](#)

`assert_length`*Check Object Length/Dimensions*

Description

Check Object Length/Dimensions

Usage

```
assert_length(x, expected, empty = FALSE)
```

```
assert_lengths(x, expected)
```

```
assert_empty(x)
```

```
assert_filled(x)
```

```
assert_dimensions(x, expected)
```

Arguments

x	An object to be checked.
expected	An appropriate expected value.
empty	A logical scalar: should empty objects be ignored?

Value

Throws an error, if any, and returns x invisibly otherwise.

Author(s)

N. Frerebeau

See Also

Other validation methods: [assert_constant\(\)](#), [assert_data](#), [assert_lower\(\)](#), [assert_names\(\)](#), [assert_numeric](#), [assert_package\(\)](#), [assert_square\(\)](#), [assert_type\(\)](#), [validate\(\)](#)

assert_lower

Check Numeric Relations

Description

Check Numeric Relations

Usage

```
assert_lower(x, y, ...)
```

```
assert_greater(x, y, ...)
```

Arguments

x, y	A numeric object to be checked.
...	Extra parameters to be passed to internal methods.

Value

Throws an error, if any, and returns `x` invisibly otherwise.

Author(s)

N. Frerebeau

See Also

Other validation methods: [assert_constant\(\)](#), [assert_data](#), [assert_length\(\)](#), [assert_names\(\)](#), [assert_numeric](#), [assert_package\(\)](#), [assert_square\(\)](#), [assert_type\(\)](#), [validate\(\)](#)

assert_names	<i>Check Object Names</i>
--------------	---------------------------

Description

Check Object Names

Usage

```
assert_names(x, expected)
```

```
assert_dimnames(x, expected)
```

```
assert_rownames(x, expected)
```

```
assert_colnames(x, expected)
```

Arguments

<code>x</code>	An object to be checked.
<code>expected</code>	An appropriate expected value.

Value

Throws an error, if any, and returns `x` invisibly otherwise.

Author(s)

N. Frerebeau

See Also

Other validation methods: [assert_constant\(\)](#), [assert_data](#), [assert_length\(\)](#), [assert_lower\(\)](#), [assert_numeric](#), [assert_package\(\)](#), [assert_square\(\)](#), [assert_type\(\)](#), [validate\(\)](#)

assert_numeric	<i>Check Numeric Values</i>
----------------	-----------------------------

Description

Check Numeric Values

Usage

```
assert_count(x, na.rm = FALSE, ...)  
assert_whole(x, na.rm = FALSE, ...)  
assert_positive(x, na.rm = FALSE, ...)  
assert_negative(x, na.rm = FALSE, ...)  
assert_odd(x, na.rm = FALSE, ...)  
assert_even(x, na.rm = FALSE, ...)
```

Arguments

x	A numeric object to be checked.
na.rm	A logical scalar: should missing values (including NaN) be omitted?
...	Extra parameters to be passed to internal methods.

Value

Throws an error, if any, and returns x invisibly otherwise.

Author(s)

N. Frerebeau

See Also

Other validation methods: [assert_constant\(\)](#), [assert_data](#), [assert_length\(\)](#), [assert_lower\(\)](#), [assert_names\(\)](#), [assert_package\(\)](#), [assert_square\(\)](#), [assert_type\(\)](#), [validate\(\)](#)

assert_package	<i>Check the Availability of a Package</i>
----------------	--

Description

Check the Availability of a Package

Usage

```
assert_package(x, ask = TRUE)
```

```
needs(x, ask = TRUE)
```

Arguments

x	A character vector naming the packages to check.
ask	A logical scalar: should the user be asked to select packages before they are downloaded and installed?

Details

`assert_package()` is designed for use inside other functions in your own package to check for the availability of a suggested package.

If the required packages are not available and R is running interactively, the user will be asked to install the packages.

`needs()` is an alias for `assert_package()`.

Value

Invisibly returns NULL.

Author(s)

N. Frerebeau

See Also

Other validation methods: [assert_constant\(\)](#), [assert_data](#), [assert_length\(\)](#), [assert_lower\(\)](#), [assert_names\(\)](#), [assert_numeric](#), [assert_square\(\)](#), [assert_type\(\)](#), [validate\(\)](#)

assert_square	<i>Check Matrix</i>
---------------	---------------------

Description

Check Matrix

Usage

```
assert_square(x)
```

```
assert_symmetric(x)
```

Arguments

x A [matrix](#) to be checked.

Value

Throw an error, if any, and returns x invisibly otherwise.

Author(s)

N. Frerebeau

See Also

Other validation methods: [assert_constant\(\)](#), [assert_data](#), [assert_length\(\)](#), [assert_lower\(\)](#), [assert_names\(\)](#), [assert_numeric](#), [assert_package\(\)](#), [assert_type\(\)](#), [validate\(\)](#)

assert_type	<i>Check Data Types</i>
-------------	-------------------------

Description

Check Data Types

Usage

```
assert_type(x, expected)
```

```
assert_scalar(x, expected)
```

```
assert_function(x)
```

Arguments

x	An object to be checked.
expected	A character string specifying the expected type. It must be one of "list", "atomic", "vector", "numeric", "integer", "double", "character" or "logical".

Value

Throws an error, if any, and returns x invisibly otherwise.

Author(s)

N. Frerebeau

See Also

Other validation methods: [assert_constant\(\)](#), [assert_data](#), [assert_length\(\)](#), [assert_lower\(\)](#), [assert_names\(\)](#), [assert_numeric](#), [assert_package\(\)](#), [assert_square\(\)](#), [validate\(\)](#)

assign

Assign a Specific Row/Column to the Column/Row Names

Description

Assign a Specific Row/Column to the Column/Row Names

Usage

```
assign_colnames(x, ...)
assign_rownames(x, ...)

## S4 method for signature 'data.frame'
assign_rownames(x, column, remove = TRUE)

## S4 method for signature 'data.frame'
assign_colnames(x, row, remove = TRUE)
```

Arguments

x	A data.frame .
...	Currently not used.
column	A length-one numeric vector specifying the column number that is to become the row names.
remove	A logical scalar: should the specified row/column be removed after making it the column/row names?
row	A length-one numeric vector specifying the row number that is to become the column names.

Value

A `data.frame`.

Author(s)

N. Frerebeau

See Also

Other data preparation tools: [append\(\)](#), [compact\(\)](#), [count\(\)](#), [detect\(\)](#), [discard\(\)](#), [get\(\)](#), [keep\(\)](#), [seek\(\)](#)

Examples

```
X <- data.frame(  
  x = 1:5,  
  y = 6:10,  
  z = LETTERS[1:5]  
)  
  
## Assign column to row names  
(Y <- assign_rownames(X, 3))  
  
## Append row names to data.frame  
(Z <- append_rownames(Y))
```

bootstrap

Bootstrap Estimation

Description

Samples randomly from the elements of object with replacement.

Usage

```
bootstrap(object, ...)  
  
## S4 method for signature 'numeric'  
bootstrap(object, do, n, ..., f = NULL)
```

Arguments

object	A numeric vector.
...	Extra arguments to be passed to do.
do	A function that takes object as an argument and returns a single numeric value.
n	A non-negative integer giving the number of bootstrap replications.
f	A function that takes a single numeric vector (the result of do) as argument.

Value

If `f` is `NULL` (the default), `bootstrap()` returns a named numeric vector with the following elements:

`original` The observed value of `do` applied to object.

`mean` The bootstrap estimate of mean of `do`.

`bias` The bootstrap estimate of bias of `do`.

`error` The bootstrap estimate of standard error of `do`.

If `f` is a function, `bootstrap()` returns the result of `f` applied to the `n` values of `do`.

Author(s)

N. Frerebeau

See Also

Other resampling methods: [jackknife\(\)](#)

Examples

```
x <- rnorm(20)

## Bootstrap
bootstrap(x, do = mean, n = 100)

## Estimate the 25th and 95th percentiles
quant <- function(x) { quantile(x, probs = c(0.25, 0.75)) }
bootstrap(x, n = 100, do = mean, f = quant)

## Get the n bootstrap values
bootstrap(x, n = 100, do = mean, f = function(x) { x })

## Jackknife
jackknife(x, do = mean) # Sample mean

## Get the leave-one-out values instead of summary
jackknife(x, do = mean, f = function(x) { x })
```

clean_whitespace

Remove Leading/Trailing Whitespace

Description

Remove Leading/Trailing Whitespace

Usage

```
clean_whitespace(x, ...)

## S4 method for signature 'data.frame'
clean_whitespace(x, which = c("both", "left", "right"), squish = TRUE)

## S4 method for signature 'matrix'
clean_whitespace(x, which = c("both", "left", "right"), squish = TRUE)
```

Arguments

x	An R object (should be a matrix or a data.frame).
...	Currently not used.
which	A character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right").
squish	A logical scalar: should all internal whitespace be replaced with a single space?

Author(s)

N. Frerebeau

See Also

[trimws\(\)](#)

Other data cleaning tools: [remove_Inf\(\)](#), [remove_NA\(\)](#), [remove_constant\(\)](#), [remove_empty\(\)](#), [remove_zero\(\)](#), [replace_Inf\(\)](#), [replace_NA\(\)](#), [replace_empty\(\)](#), [replace_zero\(\)](#)

Examples

```
x <- data.frame(
  A = c(" Both ", " Left", "Right "),
  B = 1:3
)

clean_whitespace(x, which = "both")
clean_whitespace(x, which = "left")
clean_whitespace(x, which = "right")
```

compact

Remove Empty Rows/Columns

Description

Removes empty rows/columns in an array-like object.

Usage

```
compact(x, ...)

compact_cols(x, ...)

compact_rows(x, ...)

## S4 method for signature 'ANY'
compact(x, margin = 1, na.rm = FALSE, verbose = getOption("arkhe.verbose"))

## S4 method for signature 'ANY'
compact_cols(x, na.rm = FALSE, verbose = getOption("arkhe.verbose"))

## S4 method for signature 'ANY'
compact_rows(x, na.rm = FALSE, verbose = getOption("arkhe.verbose"))
```

Arguments

x	An R object (should be a matrix or a data.frame).
...	Currently not used.
margin	A length-one numeric vector giving the subscripts which the function will be applied over (1 indicates rows, 2 indicates columns).
na.rm	A logical scalar: should NA values be stripped before the computation proceeds?
verbose	A logical scalar: should R report extra information on progress?

Details

A row/column is empty if it contains only zeros (if of type `numeric`) or zero length character strings (if of type `character`).

Author(s)

N. Frerebeau

See Also

Other data preparation tools: [append\(\)](#), [assign\(\)](#), [count\(\)](#), [detect\(\)](#), [discard\(\)](#), [get\(\)](#), [keep\(\)](#), [seek\(\)](#)

Examples

```
## Create a data.frame
X <- data.frame(A = 0, B = 1:5, C = 6, D = "", F = letters[1:5])
X

## Remove empty columns
compact(X, margin = 2)
```

concat	<i>Concatenate</i>
--------	--------------------

Description

Concatenates character vectors.

Usage

```
x %>% y
```

Arguments

x, y A [character](#) vector.

Value

A [character](#) vector.

See Also

Other utilities: [null](#)

confidence_binomial	<i>Confidence Interval for Binomial Proportions</i>
---------------------	---

Description

Computes a Wald interval for a proportion at a desired level of significance.

Usage

```
confidence_binomial(object, ...)  
  
## S4 method for signature 'numeric'  
confidence_binomial(  
  object,  
  n,  
  level = 0.95,  
  method = "wald",  
  corrected = FALSE  
)
```


Arguments

object	A numeric vector giving the number of success.
...	Currently not used.
n	A length-one numeric vector giving the number of trials.
level	A length-one numeric vector giving the confidence level. Must be a single number between 0 and 1.
method	A character string specifying the method to be used. Any unambiguous substring can be used.
corrected	A logical scalar: should continuity correction be used? Only used if method is "wald".

Value

A length-two **numeric** vector giving the lower and upper confidence limits.

Author(s)

N. Frerebeau

See Also

Other summary statistics: [confidence_mean\(\)](#), [confidence_multinomial\(\)](#), [interval_credibile\(\)](#), [interval_hdr\(\)](#)

Examples

```
## Confidence interval for a mean
x <- seq(from = -4, to = 4, by = 0.01)
y <- dnorm(x)

confidence_mean(y, type = "student")
confidence_mean(y, type = "normal")

## Confidence interval for a propotion
confidence_binomial(118, n = 236)

x <- c(35, 74, 22, 69)
confidence_multinomial(x)
```

confidence_mean

Confidence Interval for a Mean

Description

Computes a confidence interval for a mean at a desired level of significance.

Usage

```
confidence_mean(object, ...)  
  
## S4 method for signature 'numeric'  
confidence_mean(object, level = 0.95, type = c("student", "normal"))
```

Arguments

object	A numeric vector.
...	Currently not used.
level	A length-one numeric vector giving the confidence level. Must be a single number between 0 and 1.
type	A character string giving the type of confidence interval to be returned. It must be one "student" (the default) or "normal". Any unambiguous substring can be given.

Value

A length-two [numeric](#) vector giving the lower and upper confidence limits.

Author(s)

N. Frerebeau

See Also

Other summary statistics: [confidence_binomial\(\)](#), [confidence_multinomial\(\)](#), [interval_credibile\(\)](#), [interval_hdr\(\)](#)

Examples

```
## Confidence interval for a mean  
x <- seq(from = -4, to = 4, by = 0.01)  
y <- dnorm(x)  
  
confidence_mean(y, type = "student")  
confidence_mean(y, type = "normal")  
  
## Confidence interval for a propotion  
confidence_binomial(118, n = 236)  
  
x <- c(35, 74, 22, 69)  
confidence_multinomial(x)
```

`confidence_multinomial`*Confidence Interval for Multinomial Proportions*

Description

Computes a Wald interval for a proportion at a desired level of significance.

Usage

```
confidence_multinomial(object, ...)  
  
## S4 method for signature 'numeric'  
confidence_multinomial(  
  object,  
  level = 0.95,  
  method = "wald",  
  corrected = FALSE  
)
```

Arguments

<code>object</code>	A numeric vector of positive integers giving the number of occurrences of each class.
<code>...</code>	Currently not used.
<code>level</code>	A length-one numeric vector giving the confidence level. Must be a single number between 0 and 1.
<code>method</code>	A character string specifying the method to be used. Any unambiguous substring can be used.
<code>corrected</code>	A logical scalar: should continuity correction be used? Only used if method is "wald".

Value

A two column [numeric](#) matrix giving the lower and upper confidence limits.

Author(s)

N. Frerebeau

See Also

Other summary statistics: [confidence_binomial\(\)](#), [confidence_mean\(\)](#), [interval_credibile\(\)](#), [interval_hdr\(\)](#)

Examples

```
## Confidence interval for a mean
x <- seq(from = -4, to = 4, by = 0.01)
y <- dnorm(x)

confidence_mean(y, type = "student")
confidence_mean(y, type = "normal")

## Confidence interval for a propotion
confidence_binomial(118, n = 236)

x <- c(35, 74, 22, 69)
confidence_multinomial(x)
```

count

Count Values Using a Predicate

Description

Counts values by rows/columns using a predicate function.

Usage

```
count(x, ...)

## S4 method for signature 'data.frame'
count(x, f, margin = 1, negate = FALSE, na.rm = FALSE, ...)

## S4 method for signature 'matrix'
count(x, f, margin = 1, negate = FALSE, na.rm = FALSE, ...)
```

Arguments

x	An R object (should be a matrix or a data.frame).
...	Further arguments to be passed to f.
f	A predicate function .
margin	A length-one numeric vector giving the subscripts which the function will be applied over (1 indicates rows, 2 indicates columns).
negate	A logical scalar: should the negation of f be used instead of f?
na.rm	A logical scalar: should NA values be stripped before the computation proceeds?

Value

A [numeric](#) vector.

Author(s)

N. Frerebeau

See Also

Other data preparation tools: [append\(\)](#), [assign\(\)](#), [compact\(\)](#), [detect\(\)](#), [discard\(\)](#), [get\(\)](#), [keep\(\)](#), [seek\(\)](#)

Examples

```
## Create a data matrix
X <- matrix(sample(1:10, 25, TRUE), nrow = 5, ncol = 5)

## Add NA
k <- sample(1:25, 3, FALSE)
X[k] <- NA
X

## Count missing values in rows
count(X, f = is.na, margin = 1)
## Count non-missing values in columns
count(X, f = is.na, margin = 2, negate = TRUE)
```

describe

Data Description

Description

Describes an object.

Usage

```
describe(x, ...)
```

S4 method for signature 'ANY'

```
describe(x)
```

Arguments

x	An R object (should be a matrix or a data.frame).
...	Currently not used.

Value

`describe()` is called for its side-effects. Invisibly returns x.

Author(s)

N. Frerebeau

See Also

Other data summaries: [sparsity\(\)](#)

Examples

```
## Create a data matrix
X <- matrix(sample(0:9, 15, TRUE), nrow = 3, ncol = 5)

## Add NA
k <- sample(1:15, 3, FALSE)
X[k] <- NA

## Sparsity
sparsity(X)

## Quick description
describe(X)
```

detect

Find Rows/Columns Using a Predicate

Description

Finds rows/columns in an array-like object using a predicate function.

Usage

```
detect(x, ...)

## S4 method for signature 'ANY'
detect(x, f, margin = 1, negate = FALSE, all = FALSE, na.rm = FALSE, ...)
```

Arguments

<code>x</code>	An R object (should be a matrix or a data.frame).
<code>...</code>	Further arguments to be passed to <code>f</code> .
<code>f</code>	A predicate function .
<code>margin</code>	A length-one numeric vector giving the subscripts which the function will be applied over (1 indicates rows, 2 indicates columns).
<code>negate</code>	A logical scalar: should the negation of <code>f</code> be used instead of <code>f</code> ?
<code>all</code>	A logical scalar. If TRUE, only the rows/columns whose values all meet the condition defined by <code>f</code> are considered. If FALSE (the default), only rows/columns where at least one value validates the condition defined by <code>f</code> are considered.
<code>na.rm</code>	A logical scalar: should NA values be stripped before the computation proceeds?

Value

A [logical](#) vector.

Author(s)

N. Frerebeau

See Also

Other data preparation tools: [append\(\)](#), [assign\(\)](#), [compact\(\)](#), [count\(\)](#), [discard\(\)](#), [get\(\)](#), [keep\(\)](#), [seek\(\)](#)

Examples

```
## Create a data matrix
X <- matrix(sample(1:10, 25, TRUE), nrow = 5, ncol = 5)

## Add NA
k <- sample(1:25, 3, FALSE)
X[k] <- NA
X

## Find row with NA
detect(X, f = is.na, margin = 1)
## Find column without any NA
detect(X, f = is.na, margin = 2, negate = TRUE, all = TRUE)
```

discard

Remove Rows/Columns Using a Predicate

Description

Removes rows/columns in an array-like object using a predicate function.

Usage

```
discard(x, ...)

discard_cols(x, ...)

discard_rows(x, ...)

## S4 method for signature 'ANY'
discard(
  x,
  f,
  margin = 1,
  negate = FALSE,
```

```

    all = FALSE,
    na.rm = FALSE,
    verbose = getOption("arkhe.verbose"),
    ...
  )

## S4 method for signature 'ANY'
discard_rows(
  x,
  f,
  negate = FALSE,
  all = FALSE,
  na.rm = FALSE,
  verbose = getOption("arkhe.verbose"),
  ...
)

## S4 method for signature 'ANY'
discard_cols(
  x,
  f,
  negate = FALSE,
  all = FALSE,
  na.rm = FALSE,
  verbose = getOption("arkhe.verbose"),
  ...
)

```

Arguments

x	An R object (should be a matrix or a data.frame).
...	Further arguments to be passed to f.
f	A predicate function .
margin	A length-one numeric vector giving the subscripts which the function will be applied over (1 indicates rows, 2 indicates columns).
negate	A logical scalar: should the negation of f be used instead of f?
all	A logical scalar. If TRUE, only the rows/columns whose values all meet the condition defined by f are considered. If FALSE (the default), only rows/columns where at least one value validates the condition defined by f are considered.
na.rm	A logical scalar: should NA values be stripped before the computation proceeds?
verbose	A logical scalar: should R report extra information on progress?

Author(s)

N. Frerebeau

See Also

Other data preparation tools: [append\(\)](#), [assign\(\)](#), [compact\(\)](#), [count\(\)](#), [detect\(\)](#), [get\(\)](#), [keep\(\)](#), [seek\(\)](#)

Examples

```
## Create a data matrix
X <- matrix(sample(1:10, 25, TRUE), nrow = 5, ncol = 5)

## Add NA
k <- sample(1:25, 3, FALSE)
X[k] <- NA
X

## Remove row with any NA
discard(X, f = is.na, margin = 1, all = FALSE)
## Remove column with any NA
discard(X, f = is.na, margin = 2, all = FALSE)
```

get

Get Rows/Columns by Name

Description

Returns rows/columns selected by name in an array-like object.

Usage

```
get_columns(x, ...)

get_rows(x, ...)

## S4 method for signature 'data.frame'
get_columns(x, select = NULL, ...)

## S4 method for signature 'data.frame'
get_rows(x, select = NULL, ...)
```

Arguments

x	An R object (should be a matrix or a data.frame).
...	Further arguments to be passed to select.
select	A function to be applied to the row/column names (e.g. startsWith()). Must return a single integer or logical vector.

Value

An object of the same sort as x.

Author(s)

N. Frerebeau

See Also

Other data preparation tools: [append\(\)](#), [assign\(\)](#), [compact\(\)](#), [count\(\)](#), [detect\(\)](#), [discard\(\)](#), [keep\(\)](#), [seek\(\)](#)

Examples

```
## Seek columns
seek_columns(iris, select = startsWith, prefix = "Sepal")

## Get columns
x <- get_columns(iris, select = startsWith, prefix = "Sepal")
head(x)
```

interval_credible *Bayesian Credible Interval*

Description

Computes the shortest credible interval within which an unobserved parameter value falls with a particular probability.

Usage

```
interval_credible(x, ...)

## S4 method for signature 'numeric'
interval_credible(x, level = 0.95)
```

Arguments

x	A numeric vector.
...	Currently not used.
level	A length-one numeric vector giving the confidence level.

Value

A three-columns numeric [matrix](#) giving the lower and upper boundaries of the credible interval and associated probability.

Author(s)

N. Frerebeau

See Also

Other summary statistics: [confidence_binomial\(\)](#), [confidence_mean\(\)](#), [confidence_multinomial\(\)](#), [interval_hdr\(\)](#)

Examples

```
## HDR of the Old Faithful eruption times
interval_hdr(faithful$eruptions)
```

interval_hdr	<i>Highest Density Regions</i>
--------------	--------------------------------

Description

Highest Density Regions

Usage

```
interval_hdr(x, y, ...)

## S4 method for signature 'numeric,numeric'
interval_hdr(x, y, level = 0.954)

## S4 method for signature 'numeric,missing'
interval_hdr(x, level = 0.954, ...)
```

Arguments

x	A numeric vector giving the coordinates of the points where the density is estimated.
y	A numeric vector giving the estimated density values. If y is missing and x is a numeric vector, density estimates will be computed from x.
...	Further arguments to be passed to stats::density() .
level	A length-one numeric vector giving the confidence level.

Value

A three-columns numeric **matrix** giving the lower and upper boundaries of the HPD interval and associated probabilities.

Author(s)

N. Frerebeau

References

Hyndman, R. J. (1996). Computing and graphing highest density regions. *American Statistician*, 50: 120-126. doi:10.2307/2684423.

See Also

Other summary statistics: [confidence_binomial\(\)](#), [confidence_mean\(\)](#), [confidence_multinomial\(\)](#), [interval_credible\(\)](#)

Examples

```
## HDR of the Old Faithful eruption times
interval_hdr(faithful$eruptions)
```

is_scalar	<i>Scalar Type Predicates</i>
-----------	-------------------------------

Description

Scalar Type Predicates

Usage

```
is_scalar_list(x)
is_scalar_atomic(x)
is_scalar_vector(x)
is_scalar_numeric(x)
is_scalar_integer(x)
is_scalar_double(x)
is_scalar_character(x)
is_scalar_logical(x)
```

Arguments

x An object to be tested.

Value

A [logical](#) scalar.

See Also

Other predicates: [predicate-matrix](#), [predicate-numeric](#), [predicate-trend](#), [predicate-type](#), [predicate-utils](#)

`jackknife`*Jackknife Estimation*

Description

Jackknife Estimation

Usage`jackknife(object, ...)`

```
## S4 method for signature 'numeric'
jackknife(object, do, ..., f = NULL)
```

Arguments

<code>object</code>	A numeric vector.
<code>...</code>	Extra arguments to be passed to <code>do</code> .
<code>do</code>	A function that takes <code>object</code> as an argument and returns a single numeric value.
<code>f</code>	A function that takes a single numeric vector (the leave-one-out values of <code>do</code>) as argument.

Value

If `f` is `NULL` (the default), `jackknife()` returns a named numeric vector with the following elements:

`original` The observed value of `do` applied to `object`.

`mean` The jackknife estimate of mean of `do`.

`bias` The jackknife estimate of bias of `do`.

`error` The jackknife estimate of standard error of `do`.

If `f` is a function, `jackknife()` returns the result of `f` applied to the leave-one-out values of `do`.

Author(s)

N. Frerebeau

See AlsoOther resampling methods: [bootstrap\(\)](#)

Examples

```

x <- rnorm(20)

## Bootstrap
bootstrap(x, do = mean, n = 100)

## Estimate the 25th and 95th percentiles
quant <- function(x) { quantile(x, probs = c(0.25, 0.75)) }
bootstrap(x, n = 100, do = mean, f = quant)

## Get the n bootstrap values
bootstrap(x, n = 100, do = mean, f = function(x) { x })

## Jackknife
jackknife(x, do = mean) # Sample mean

## Get the leave-one-out values instead of summary
jackknife(x, do = mean, f = function(x) { x })

```

keep	<i>Keep Rows/Columns Using a Predicate</i>
------	--

Description

Keeps rows/columns in an array-like object using a predicate function.

Usage

```

keep(x, ...)

keep_cols(x, ...)

keep_rows(x, ...)

## S4 method for signature 'ANY'
keep(
  x,
  f,
  margin = 1,
  negate = FALSE,
  all = FALSE,
  na.rm = FALSE,
  verbose = getOption("arkhe.verbose"),
  ...
)

## S4 method for signature 'ANY'
keep_rows(

```

```

    x,
    f,
    negate = FALSE,
    all = FALSE,
    na.rm = FALSE,
    verbose = getOption("arkhe.verbose"),
    ...
)

## S4 method for signature 'ANY'
keep_cols(
  x,
  f,
  negate = FALSE,
  all = FALSE,
  na.rm = FALSE,
  verbose = getOption("arkhe.verbose"),
  ...
)

```

Arguments

<code>x</code>	An R object (should be a matrix or a data.frame).
<code>...</code>	Further arguments to be passed to <code>f</code> .
<code>f</code>	A predicate function .
<code>margin</code>	A length-one numeric vector giving the subscripts which the function will be applied over (1 indicates rows, 2 indicates columns).
<code>negate</code>	A logical scalar: should the negation of <code>f</code> be used instead of <code>f</code> ?
<code>all</code>	A logical scalar. If TRUE, only the rows/columns whose values all meet the condition defined by <code>f</code> are considered. If FALSE (the default), only rows/columns where at least one value validates the condition defined by <code>f</code> are considered.
<code>na.rm</code>	A logical scalar: should NA values be stripped before the computation proceeds?
<code>verbose</code>	A logical scalar: should R report extra information on progress?

Author(s)

N. Frerebeau

See Also

Other data preparation tools: [append\(\)](#), [assign\(\)](#), [compact\(\)](#), [count\(\)](#), [detect\(\)](#), [discard\(\)](#), [get\(\)](#), [seek\(\)](#)

Examples

```
## Create a data matrix
X <- matrix(sample(1:10, 25, TRUE), nrow = 5, ncol = 5)

## Add NA
k <- sample(1:25, 3, FALSE)
X[k] <- NA
X

## Keep row without any NA
keep(X, f = is.na, margin = 1, negate = TRUE, all = TRUE)
## Keep row without any NA
keep(X, f = is.na, margin = 2, negate = TRUE, all = TRUE)
```

math_gcd

Greatest Common Divisor

Description

Computes the greatest common divisor (GCD) of two integer using the Euclidean algorithm.

Usage

```
math_gcd(x, y)

## S4 method for signature 'numeric,numeric'
math_gcd(x, y)
```

Arguments

x, y A [numeric](#) vector.

Value

A [numeric](#) vector.

Author(s)

N. Frerebeau

See Also

Other mathematic functions: [math_lcm\(\)](#)

math_lcm	<i>Least Common Multiple</i>
----------	------------------------------

Description

Computes the lowest common multiple of the denominators of a set of fractions.

Usage

```
math_lcm(x, y)
```

```
## S4 method for signature 'numeric,numeric'
```

```
math_lcm(x, y)
```

Arguments

x, y A [numeric](#) vector.

Value

A [numeric](#) vector.

Author(s)

N. Frerebeau

See Also

Other mathematic functions: [math_gcd\(\)](#)

null	<i>Default value for NULL</i>
------	-------------------------------

Description

Replaces NULL with a default value.

Usage

```
x %||% y
```

Arguments

x, y An object.

Value

If `x` is `NULL`, returns `y`; otherwise returns `x`.

See Also

Other utilities: [concat](#)

palette_color	<i>Color Mapping</i>
---------------	----------------------

Description

Color Mapping

Usage

```
palette_color_continuous(x, palette = NULL, domain = NULL, na = "#DDDDDD", ...)
```

```
palette_color_discrete(
  x,
  palette = NULL,
  domain = NULL,
  ordered = FALSE,
  na = "#DDDDDD",
  ...
)
```

Arguments

<code>x</code>	A vector of values.
<code>palette</code>	A vector of colors or a color function that values will be mapped to (see grDevices::colorRamp()).
<code>domain</code>	A numeric range or categorical data (according to <code>x</code>).
<code>na</code>	The color to return for NA values.
<code>...</code>	Currently not used.
<code>ordered</code>	A logical scalar: should the levels be treated as already in the correct order?

Value

A **character** vector of colors.

See Also

Other palettes: [palette_shape\(\)](#), [palette_size\(\)](#)

palette_shape	<i>Symbol Mapping</i>
---------------	-----------------------

Description

Symbol Mapping

Usage

```
palette_shape(x, palette = NULL, domain = NULL, ordered = FALSE, ...)
```

Arguments

x	A vector of categorical values.
palette	A vector of symbols.
domain	A vector of categorical data.
ordered	A logical scalar: should the levels be treated as already in the correct order?
...	Currently not used.

Value

A vector of symbols.

See Also

Other palettes: [palette_color](#), [palette_size\(\)](#)

palette_size	<i>Symbol Size Mapping</i>
--------------	----------------------------

Description

Symbol Size Mapping

Usage

```
palette_size(x, palette = NULL, domain = NULL, ...)
```

Arguments

x	A vector of categorical values.
palette	A numeric range.
domain	A vector of categorical data.
...	Currently not used.

Value

A [numeric](#) vector giving the amount by which plotting text and symbols should be magnified relative to the default (see `par("cex")`).

See Also

Other palettes: [palette_color](#), [palette_shape](#)()

predicate-matrix *Matrix Predicates*

Description

- `is_square()` checks if a matrix is square.
- `is_symmetric()` checks if a matrix is symmetric.

Usage

```
is_square(x)
```

```
is_symmetric(x)
```

Arguments

x A [matrix](#) to be tested.

Value

A [logical](#) scalar.

See Also

Other predicates: [is_scalar](#), [predicate-numeric](#), [predicate-trend](#), [predicate-type](#), [predicate-utils](#)

Description

Check numeric objects:

- `is_zero()` checks if an object contains only zeros.
- `is_odd()` and `is_even()` check if a number is odd or even, respectively.
- `is_positive()` and `is_negative` check if an object contains only (strictly) positive or negative numbers.
- `is_whole()` checks if an object only contains whole numbers.

Usage

```
is_zero(x, tolerance = sqrt(.Machine$double.eps), ...)
```

```
is_odd(x, ...)
```

```
is_even(x, ...)
```

```
is_positive(x, strict = FALSE, ...)
```

```
is_negative(x, strict = FALSE, ...)
```

```
is_whole(x, tolerance = sqrt(.Machine$double.eps), ...)
```

Arguments

<code>x</code>	A numeric object to be tested.
<code>tolerance</code>	A numeric scalar giving the tolerance to check within.
<code>...</code>	Currently not used.
<code>strict</code>	A logical scalar: should strict inequality be used?

Value

A [logical](#) vector.

See Also

Other predicates: [is_scalar](#), [predicate-matrix](#), [predicate-trend](#), [predicate-type](#), [predicate-utils](#)

predicate-trend *Numeric Trend Predicates*

Description

Check numeric objects:

- `is_constant()` checks for equality among all elements of a vector.
- `is_increasing()` and `is_decreasing()` check if a sequence of numbers is monotonically increasing or decreasing, respectively.

Usage

```
is_constant(x, tolerance = sqrt(.Machine$double.eps), na.rm = FALSE)
```

```
is_increasing(x, na.rm = FALSE)
```

```
is_decreasing(x, na.rm = FALSE)
```

```
is_greater(x, y, strict = FALSE, na.rm = FALSE)
```

```
is_lower(x, y, strict = FALSE, na.rm = FALSE)
```

Arguments

<code>x, y</code>	A numeric object to be tested.
<code>tolerance</code>	A numeric scalar giving the tolerance to check within.
<code>na.rm</code>	A logical scalar: should missing values (including NaN) be omitted?
<code>strict</code>	A logical scalar: should strict inequality be used?

Value

A **logical** scalar.

See Also

Other predicates: [is_scalar](#), [predicate-matrix](#), [predicate-numeric](#), [predicate-type](#), [predicate-utils](#)

predicate-type	<i>Type Predicates</i>
----------------	------------------------

Description

Type Predicates

Usage

`is_list(x)`

`is_atomic(x)`

`is_vector(x)`

`is_numeric(x)`

`is_integer(x)`

`is_double(x)`

`is_character(x)`

`is_logical(x)`

`is_error(x)`

`is_warning(x)`

`is_message(x)`

Arguments

`x` An object to be tested.

Value

A [logical](#) scalar.

See Also

Other predicates: [is_scalar](#), [predicate-matrix](#), [predicate-numeric](#), [predicate-trend](#), [predicate-utils](#)

Description

- `is_empty()` checks if an object is empty (any zero-length dimensions).
- `has_length()` checks how long is an object.
- `has_names()` checks if an object is named.
- `has_duplicates()` checks if an object has duplicated elements.
- `has_missing()` and `has_infinite()` check if an object contains missing or infinite values.

Usage

```
has_length(x, n = NULL)
```

```
is_empty(x)
```

```
has_names(x, names = NULL)
```

```
has_missing(x)
```

```
has_infinite(x)
```

```
is_unique(x, tolerance = sqrt(.Machine$double.eps), na.rm = FALSE)
```

```
has_duplicates(x)
```

Arguments

<code>x</code>	A vector to be tested.
<code>n</code>	A length-one numeric vector specifying the length to test <code>x</code> with. If <code>NULL</code> , returns <code>TRUE</code> if <code>x</code> has length greater than zero, and <code>FALSE</code> otherwise.
<code>names</code>	A character vector specifying the names to test <code>x</code> with. If <code>NULL</code> , returns <code>TRUE</code> if <code>x</code> has names, and <code>FALSE</code> otherwise.
<code>tolerance</code>	A numeric scalar giving the tolerance to check within (for numeric vector).
<code>na.rm</code>	A logical scalar: should missing values (including <code>NaN</code>) be omitted?

Value

A [logical](#) scalar.

See Also

Other predicates: [is_scalar](#), [predicate-matrix](#), [predicate-numeric](#), [predicate-trend](#), [predicate-type](#)

remove_constant	<i>Remove Constant Columns</i>
-----------------	--------------------------------

Description

Remove Constant Columns

Usage

```
remove_constant(x, ...)  
  
## S4 method for signature 'ANY'  
remove_constant(x, na.rm = FALSE, verbose = getOption("arkhe.verbose"))
```

Arguments

x	An R object (should be a matrix or a data.frame).
...	Currently not used.
na.rm	A logical scalar: should NA values be stripped before the computation proceeds?
verbose	A logical scalar: should R report extra information on progress?

Author(s)

N. Frerebeau

See Also

Other data cleaning tools: [clean_whitespace\(\)](#), [remove_Inf\(\)](#), [remove_NA\(\)](#), [remove_empty\(\)](#), [remove_zero\(\)](#), [replace_Inf\(\)](#), [replace_NA\(\)](#), [replace_empty\(\)](#), [replace_zero\(\)](#)

Examples

```
## Create a data.frame  
X <- data.frame(A = 1, B = 1:3)  
X  
  
remove_constant(X)  
  
## Add NA  
X[1, 1] <- NA  
remove_constant(X)  
remove_constant(X, na.rm = TRUE)
```

remove_empty	<i>Remove Rows/Columns with Empty String</i>
--------------	--

Description

Removes rows/columns that contain empty strings.

Usage

```
remove_empty(x, ...)
```

```
## S4 method for signature 'ANY'
```

```
remove_empty(x, margin = 1, all = FALSE, verbose = getOption("arkhe.verbose"))
```

Arguments

x	An R object (should be a matrix or a data.frame).
...	Currently not used.
margin	A length-one numeric vector giving the subscripts which the function will be applied over (1 indicates rows, 2 indicates columns).
all	A logical scalar. If TRUE, only the rows/columns whose values all meet the condition defined by f are considered. If FALSE (the default), only rows/columns where at least one value validates the condition defined by f are considered.
verbose	A logical scalar: should R report extra information on progress?

Author(s)

N. Frerebeau

See Also

Other data cleaning tools: [clean_whitespace\(\)](#), [remove_Inf\(\)](#), [remove_NA\(\)](#), [remove_constant\(\)](#), [remove_zero\(\)](#), [replace_Inf\(\)](#), [replace_NA\(\)](#), [replace_empty\(\)](#), [replace_zero\(\)](#)

Examples

```
## Create a data matrix
X <- matrix(sample(LETTERS, 25, TRUE), nrow = 5, ncol = 5)

## Add empty string
k <- sample(1:25, 3, FALSE)
X[k] <- ""
X

## Remove rows with empty strings
remove_empty(X, margin = 1)
```

```
## Replace empty strings
replace_empty(X, value = "XXX")
```

remove_Inf	<i>Remove Rows/Columns with Infinite Values</i>
------------	---

Description

Removes rows/columns that contain [infinite values](#).

Usage

```
remove_Inf(x, ...)

## S4 method for signature 'ANY'
remove_Inf(x, margin = 1, all = FALSE, verbose = getOption("arkhe.verbose"))
```

Arguments

x	An R object (should be a matrix or a data.frame).
...	Currently not used.
margin	A length-one numeric vector giving the subscripts which the function will be applied over (1 indicates rows, 2 indicates columns).
all	A logical scalar. If TRUE, only the rows/columns whose values all meet the condition defined by f are considered. If FALSE (the default), only rows/columns where at least one value validates the condition defined by f are considered.
verbose	A logical scalar: should R report extra information on progress?

Author(s)

N. Frerebeau

See Also

Other data cleaning tools: [clean_whitespace\(\)](#), [remove_NA\(\)](#), [remove_constant\(\)](#), [remove_empty\(\)](#), [remove_zero\(\)](#), [replace_Inf\(\)](#), [replace_NA\(\)](#), [replace_empty\(\)](#), [replace_zero\(\)](#)

Examples

```
## Create a data matrix
X <- matrix(sample(1:10, 25, TRUE), nrow = 5, ncol = 5)

## Add Inf
k <- sample(1:25, 3, FALSE)
X[k] <- Inf
X
```

```
## Remove rows with Inf
remove_Inf(X, margin = 1)

## Replace Inf with zeros
replace_Inf(X, value = 0)
```

remove_NA

Remove Rows/Columns with Missing Values

Description

Removes rows/columns that contain [missing values](#).

Usage

```
remove_NA(x, ...)

## S4 method for signature 'ANY'
remove_NA(x, margin = 1, all = FALSE, verbose = getOption("arkhe.verbose"))
```

Arguments

x	An R object (should be a matrix or a data.frame).
...	Currently not used.
margin	A length-one numeric vector giving the subscripts which the function will be applied over (1 indicates rows, 2 indicates columns).
all	A logical scalar. If TRUE, only the rows/columns whose values all meet the condition defined by f are considered. If FALSE (the default), only rows/columns where at least one value validates the condition defined by f are considered.
verbose	A logical scalar: should R report extra information on progress?

Author(s)

N. Frerebeau

See Also

Other data cleaning tools: [clean_whitespace\(\)](#), [remove_Inf\(\)](#), [remove_constant\(\)](#), [remove_empty\(\)](#), [remove_zero\(\)](#), [replace_Inf\(\)](#), [replace_NA\(\)](#), [replace_empty\(\)](#), [replace_zero\(\)](#)

Examples

```
## Create a data matrix
X <- matrix(sample(1:10, 25, TRUE), nrow = 5, ncol = 5)

## Add NA
k <- sample(1:25, 3, FALSE)
X[k] <- NA
X

## Remove rows with NA
remove_NA(X, margin = 1)

## Replace NA with zeros
replace_NA(X, value = 0)
```

remove_zero	<i>Remove Rows/Columns with Zeros</i>
-------------	---------------------------------------

Description

Removes rows/columns that contain zeros.

Usage

```
remove_zero(x, ...)
```

S4 method for signature 'ANY'

```
remove_zero(x, margin = 1, all = FALSE, verbose = getOption("arkhe.verbose"))
```

Arguments

x	An R object (should be a matrix or a data.frame).
...	Currently not used.
margin	A length-one numeric vector giving the subscripts which the function will be applied over (1 indicates rows, 2 indicates columns).
all	A logical scalar. If TRUE, only the rows/columns whose values all meet the condition defined by f are considered. If FALSE (the default), only rows/columns where at least one value validates the condition defined by f are considered.
verbose	A logical scalar: should R report extra information on progress?

Author(s)

N. Frerebeau

See Also

Other data cleaning tools: [clean_whitespace\(\)](#), [remove_Inf\(\)](#), [remove_NA\(\)](#), [remove_constant\(\)](#), [remove_empty\(\)](#), [replace_Inf\(\)](#), [replace_NA\(\)](#), [replace_empty\(\)](#), [replace_zero\(\)](#)

Examples

```
## Create a data matrix
X <- matrix(sample(1:10, 25, TRUE), nrow = 5, ncol = 5)

## Add zero
k <- sample(1:25, 3, FALSE)
X[k] <- 0
X

## Remove rows with zero
remove_zero(X, margin = 1)

## Replace zero
replace_zero(X, value = 1)
```

replace_empty	<i>Replace Empty String</i>
---------------	-----------------------------

Description

Replaces empty strings.

Usage

```
replace_empty(x, ...)
```

S4 method for signature 'matrix'
replace_empty(x, value)

S4 method for signature 'data.frame'
replace_empty(x, value)

Arguments

x	An R object (should be a matrix or a data.frame).
...	Currently not used.
value	A possible replacement value.

Author(s)

N. Frerebeau

See Also

Other data cleaning tools: [clean_whitespace\(\)](#), [remove_Inf\(\)](#), [remove_NA\(\)](#), [remove_constant\(\)](#), [remove_empty\(\)](#), [remove_zero\(\)](#), [replace_Inf\(\)](#), [replace_NA\(\)](#), [replace_zero\(\)](#)

Examples

```
## Create a data matrix
X <- matrix(sample(LETTERS, 25, TRUE), nrow = 5, ncol = 5)

## Add empty string
k <- sample(1:25, 3, FALSE)
X[k] <- ""
X

## Remove rows with empty strings
remove_empty(X, margin = 1)

## Replace empty strings
replace_empty(X, value = "XXX")
```

replace_Inf	<i>Replace Infinite Values</i>
-------------	--------------------------------

Description

Replaces [infinite values](#) values.

Usage

```
replace_Inf(x, ...)
```

S4 method for signature 'matrix'

```
replace_Inf(x, value = 0)
```

S4 method for signature 'data.frame'

```
replace_Inf(x, value = 0)
```

Arguments

x	An R object (should be a matrix or a data.frame).
...	Currently not used.
value	A possible replacement value.

Author(s)

N. Frerebeau

See Also

Other data cleaning tools: [clean_whitespace\(\)](#), [remove_Inf\(\)](#), [remove_NA\(\)](#), [remove_constant\(\)](#), [remove_empty\(\)](#), [remove_zero\(\)](#), [replace_NA\(\)](#), [replace_empty\(\)](#), [replace_zero\(\)](#)

Examples

```
## Create a data matrix
X <- matrix(sample(1:10, 25, TRUE), nrow = 5, ncol = 5)

## Add Inf
k <- sample(1:25, 3, FALSE)
X[k] <- Inf
X

## Remove rows with Inf
remove_Inf(X, margin = 1)

## Replace Inf with zeros
replace_Inf(X, value = 0)
```

replace_NA	<i>Replace Missing Values</i>
------------	-------------------------------

Description

Replaces [missing values](#) values.

Usage

```
replace_NA(x, ...)
```

S4 method for signature 'matrix'
replace_NA(x, value = 0)

S4 method for signature 'data.frame'
replace_NA(x, value = 0)

Arguments

x	An R object (should be a matrix or a data.frame).
...	Currently not used.
value	A possible replacement value.

Author(s)

N. Frerebeau

See Also

Other data cleaning tools: [clean_whitespace\(\)](#), [remove_Inf\(\)](#), [remove_NA\(\)](#), [remove_constant\(\)](#), [remove_empty\(\)](#), [remove_zero\(\)](#), [replace_Inf\(\)](#), [replace_empty\(\)](#), [replace_zero\(\)](#)

Examples

```
## Create a data matrix
X <- matrix(sample(1:10, 25, TRUE), nrow = 5, ncol = 5)

## Add NA
k <- sample(1:25, 3, FALSE)
X[k] <- NA
X

## Remove rows with NA
remove_NA(X, margin = 1)

## Replace NA with zeros
replace_NA(X, value = 0)
```

replace_zero	<i>Replace Zeros</i>
--------------	----------------------

Description

Replaces zeros.

Usage

```
replace_zero(x, ...)
```

S4 method for signature 'matrix'
replace_zero(x, value)

S4 method for signature 'data.frame'
replace_zero(x, value)

Arguments

x	An R object (should be a matrix or a data.frame).
...	Currently not used.
value	A possible replacement value.

Author(s)

N. Frerebeau

See Also

Other data cleaning tools: [clean_whitespace\(\)](#), [remove_Inf\(\)](#), [remove_NA\(\)](#), [remove_constant\(\)](#), [remove_empty\(\)](#), [remove_zero\(\)](#), [replace_Inf\(\)](#), [replace_NA\(\)](#), [replace_empty\(\)](#)

Examples

```
## Create a data matrix
X <- matrix(sample(1:10, 25, TRUE), nrow = 5, ncol = 5)

## Add zero
k <- sample(1:25, 3, FALSE)
X[k] <- 0
X

## Remove rows with zero
remove_zero(X, margin = 1)

## Replace zero
replace_zero(X, value = 1)
```

scale_range

Rescale Continuous Vector

Description

Rescales continuous vector to have specified minimum and maximum.

Usage

```
scale_range(x, to = c(0, 1), from = range(x, finite = TRUE))
```

Arguments

x	A numeric vector.
to	A length-two numeric vector specifying the output range.
from	A length-two numeric vector specifying the input range.

Value

A [numeric](#) vector.

Note

For internal use only.

seek	<i>Search Rows/Columns by Name</i>
------	------------------------------------

Description

Searches rows/columns by name in an array-like object.

Usage

```
seek_columns(x, ...)  
  
seek_rows(x, ...)  
  
## S4 method for signature 'data.frame'  
seek_rows(x, select = NULL, ...)  
  
## S4 method for signature 'data.frame'  
seek_columns(x, select = NULL, ...)
```

Arguments

x	An R object (should be a matrix or a data.frame).
...	Further arguments to be passed to select.
select	A function to be applied to the row/column names (e.g. startsWith()). Must return a single integer or logical vector.

Value

An [integer](#) vector or NULL.

Author(s)

N. Frerebeau

See Also

Other data preparation tools: [append\(\)](#), [assign\(\)](#), [compact\(\)](#), [count\(\)](#), [detect\(\)](#), [discard\(\)](#), [get\(\)](#), [keep\(\)](#)

Examples

```
## Seek columns  
seek_columns(iris, select = startsWith, prefix = "Sepal")  
  
## Get columns  
x <- get_columns(iris, select = startsWith, prefix = "Sepal")  
head(x)
```

`sparsity`*Sparsity*

Description

Computes data sparsity (proportion of zeros).

Usage

```
sparsity(x, ...)  
  
## S4 method for signature 'matrix'  
sparsity(x, count = FALSE)  
  
## S4 method for signature 'data.frame'  
sparsity(x, count = FALSE)
```

Arguments

<code>x</code>	An R object (should be a matrix or a data.frame).
<code>...</code>	Currently not used.
<code>count</code>	A logical scalar: should a count be returned instead of a proportion?

Details

If `x` is a `data.frame`, sparsity is computed on numeric variables only.

Value

A length-one [numeric](#) vector.

Author(s)

N. Frerebeau

See Also

Other data summaries: [describe\(\)](#)

Examples

```
## Create a data matrix  
X <- matrix(sample(0:9, 15, TRUE), nrow = 3, ncol = 5)  
  
## Add NA  
k <- sample(1:15, 3, FALSE)  
X[k] <- NA
```

```
## Sparsity
sparsity(X)

## Quick description
describe(X)
```

validate	<i>Validate a Condition</i>
----------	-----------------------------

Description

Validate a Condition

Usage

```
validate(expr)
```

Arguments

expr An object to be evaluated.

Value

Returns NULL on success, otherwise returns the error as a string.

Author(s)

N. Frerebeau

See Also

Other validation methods: [assert_constant\(\)](#), [assert_data](#), [assert_length\(\)](#), [assert_lower\(\)](#), [assert_names\(\)](#), [assert_numeric](#), [assert_package\(\)](#), [assert_square\(\)](#), [assert_type\(\)](#)

Index

- * **data cleaning tools**
 - clean_whitespace, 13
 - remove_constant, 41
 - remove_empty, 42
 - remove_Inf, 43
 - remove_NA, 44
 - remove_zero, 45
 - replace_empty, 46
 - replace_Inf, 47
 - replace_NA, 48
 - replace_zero, 49
- * **data preparation tools**
 - append, 3
 - assign, 11
 - compact, 14
 - count, 20
 - detect, 22
 - discard, 23
 - get, 25
 - keep, 30
 - seek, 51
- * **data summaries**
 - describe, 21
 - sparsity, 52
- * **data transformation tools**
 - scale_range, 50
- * **mathematic functions**
 - math_gcd, 32
 - math_lcm, 33
- * **palettes**
 - palette_color, 34
 - palette_shape, 35
 - palette_size, 35
- * **predicates**
 - is_scalar, 28
 - predicate-matrix, 36
 - predicate-numeric, 37
 - predicate-trend, 38
 - predicate-type, 39
 - predicate-utils, 40
- * **resampling methods**
 - bootstrap, 12
 - jackknife, 29
- * **summary statistics**
 - confidence_binomial, 16
 - confidence_mean, 17
 - confidence_multinomial, 19
 - interval_credible, 26
 - interval_hdr, 27
- * **utilities**
 - concat, 16
 - null, 33
- * **validation methods**
 - assert_constant, 4
 - assert_data, 5
 - assert_length, 5
 - assert_lower, 6
 - assert_names, 7
 - assert_numeric, 8
 - assert_package, 9
 - assert_square, 10
 - assert_type, 10
 - validate, 53
- %+% (concat), 16
- append, 3, 12, 15, 21, 23, 25, 26, 31, 51
- append_rownames (append), 3
- append_rownames, data.frame-method (append), 3
- append_rownames-method (append), 3
- assert_colnames (assert_names), 7
- assert_constant, 4, 5–11, 53
- assert_count (assert_numeric), 8
- assert_data, 4, 5, 6–11, 53
- assert_decreasing (assert_constant), 4
- assert_dimensions (assert_length), 5
- assert_dimnames (assert_names), 7
- assert_empty (assert_length), 5
- assert_even (assert_numeric), 8

- assert_filled(assert_length), 5
- assert_function(assert_type), 10
- assert_greater(assert_lower), 6
- assert_increasing(assert_constant), 4
- assert_infinite(assert_data), 5
- assert_length, 4, 5, 5, 7–11, 53
- assert_lengths(assert_length), 5
- assert_lower, 4–6, 6, 7–11, 53
- assert_missing(assert_data), 5
- assert_names, 4–7, 7, 8–11, 53
- assert_negative(assert_numeric), 8
- assert_numeric, 4–7, 8, 9–11, 53
- assert_odd(assert_numeric), 8
- assert_package, 4–8, 9, 10, 11, 53
- assert_positive(assert_numeric), 8
- assert_rownames(assert_names), 7
- assert_scalar(assert_type), 10
- assert_square, 4–9, 10, 11, 53
- assert_symmetric(assert_square), 10
- assert_type, 4–10, 10, 53
- assert_unique(assert_data), 5
- assert_whole(assert_numeric), 8
- assign, 3, 11, 15, 21, 23, 25, 26, 31, 51
- assign_colnames(assign), 11
- assign_colnames,data.frame-method
(assign), 11
- assign_colnames-method(assign), 11
- assign_rownames(assign), 11
- assign_rownames,data.frame-method
(assign), 11
- assign_rownames-method(assign), 11

- bootstrap, 12, 29
- bootstrap,numeric-method(bootstrap), 12
- bootstrap-method(bootstrap), 12

- character, 3, 9, 11, 14, 16–19, 34, 40
- clean_whitespace, 13, 41–49
- clean_whitespace,data.frame-method
(clean_whitespace), 13
- clean_whitespace,matrix-method
(clean_whitespace), 13
- clean_whitespace-method
(clean_whitespace), 13
- compact, 3, 12, 14, 21, 23, 25, 26, 31, 51
- compact,ANY-method(compact), 14
- compact-method(compact), 14
- compact_cols(compact), 14
- compact_cols,ANY-method(compact), 14
- compact_cols-method(compact), 14
- compact_rows(compact), 14
- compact_rows,ANY-method(compact), 14
- compact_rows-method(compact), 14
- concat, 16, 34
- confidence_binomial, 16, 18, 19, 27, 28
- confidence_binomial,numeric-method
(confidence_binomial), 16
- confidence_binomial-method
(confidence_binomial), 16
- confidence_mean, 17, 17, 19, 27, 28
- confidence_mean,numeric-method
(confidence_mean), 17
- confidence_mean-method
(confidence_mean), 17
- confidence_multinomial, 17, 18, 19, 27, 28
- confidence_multinomial,numeric-method
(confidence_multinomial), 19
- confidence_multinomial-method
(confidence_multinomial), 19
- count, 3, 12, 15, 20, 23, 25, 26, 31, 51
- count,data.frame-method(count), 20
- count,matrix-method(count), 20
- count-method(count), 20

- data.frame, 3, 11, 12, 14, 15, 20–22, 24, 25,
31, 41–49, 51, 52
- describe, 21, 52
- describe,ANY-method(describe), 21
- describe-method(describe), 21
- detect, 3, 12, 15, 21, 22, 25, 26, 31, 51
- detect,ANY-method(detect), 22
- detect-method(detect), 22
- discard, 3, 12, 15, 21, 23, 23, 26, 31, 51
- discard,ANY-method(discard), 23
- discard-method(discard), 23
- discard_cols(discard), 23
- discard_cols,ANY-method(discard), 23
- discard_cols-method(discard), 23
- discard_rows(discard), 23
- discard_rows,ANY-method(discard), 23
- discard_rows-method(discard), 23

- function, 12, 20, 22, 24, 25, 29, 31, 51
- get, 3, 12, 15, 21, 23, 25, 25, 31, 51
- get_columns(get), 25
- get_columns,data.frame-method(get), 25
- get_columns-method(get), 25

- get_rows (get), 25
- get_rows, data.frame-method (get), 25
- get_rows-method (get), 25
- grDevices::colorRamp(), 34
- has_duplicates (predicate-utils), 40
- has_infinite (predicate-utils), 40
- has_length (predicate-utils), 40
- has_missing (predicate-utils), 40
- has_names (predicate-utils), 40
- infinite values, 43, 47
- integer, 12, 51
- interval_credible, 17–19, 26, 28
- interval_credible, numeric-method (interval_credible), 26
- interval_credible-method (interval_credible), 26
- interval_hdr, 17–19, 27, 27
- interval_hdr, numeric, missing-method (interval_hdr), 27
- interval_hdr, numeric, numeric-method (interval_hdr), 27
- interval_hdr-method (interval_hdr), 27
- is_atomic (predicate-type), 39
- is_character (predicate-type), 39
- is_constant (predicate-trend), 38
- is_decreasing (predicate-trend), 38
- is_double (predicate-type), 39
- is_empty (predicate-utils), 40
- is_error (predicate-type), 39
- is_even (predicate-numeric), 37
- is_greater (predicate-trend), 38
- is_increasing (predicate-trend), 38
- is_integer (predicate-type), 39
- is_list (predicate-type), 39
- is_logical (predicate-type), 39
- is_lower (predicate-trend), 38
- is_message (predicate-type), 39
- is_negative (predicate-numeric), 37
- is_numeric (predicate-type), 39
- is_odd (predicate-numeric), 37
- is_positive (predicate-numeric), 37
- is_scalar, 28, 36–40
- is_scalar_atomic (is_scalar), 28
- is_scalar_character (is_scalar), 28
- is_scalar_double (is_scalar), 28
- is_scalar_integer (is_scalar), 28
- is_scalar_list (is_scalar), 28
- is_scalar_logical (is_scalar), 28
- is_scalar_numeric (is_scalar), 28
- is_scalar_vector (is_scalar), 28
- is_square (predicate-matrix), 36
- is_symmetric (predicate-matrix), 36
- is_unique (predicate-utils), 40
- is_vector (predicate-type), 39
- is_warning (predicate-type), 39
- is_whole (predicate-numeric), 37
- is_zero (predicate-numeric), 37
- jackknife, 13, 29
- jackknife, numeric-method (jackknife), 29
- jackknife-method (jackknife), 29
- keep, 3, 12, 15, 21, 23, 25, 26, 30, 51
- keep, ANY-method (keep), 30
- keep-method (keep), 30
- keep_cols (keep), 30
- keep_cols, ANY-method (keep), 30
- keep_cols-method (keep), 30
- keep_rows (keep), 30
- keep_rows, ANY-method (keep), 30
- keep_rows-method (keep), 30
- logical, 3, 6, 8, 9, 11, 14, 15, 17, 19, 20, 22–24, 28, 31, 34–45, 52
- math_gcd, 32, 33
- math_gcd, numeric, numeric-method (math_gcd), 32
- math_gcd-method (math_gcd), 32
- math_lcm, 32, 33
- math_lcm, numeric, numeric-method (math_lcm), 33
- math_lcm-method (math_lcm), 33
- matrix, 10, 14, 15, 20–22, 24–27, 31, 36, 41–49, 51, 52
- missing (remove_NA), 44
- missing values, 44, 48
- needs (assert_package), 9
- null, 16, 33
- numeric, 3, 4, 6, 8, 11, 12, 15, 17–20, 22, 24, 26, 27, 29, 31–38, 40, 42–45, 50, 52
- palette_color, 34, 35, 36
- palette_color_continuous (palette_color), 34

- palette_color_discrete (palette_color),
34
- palette_shape, 34, 35, 36
- palette_size, 34, 35, 35
- par(cex), 36
- predicate-matrix, 36
- predicate-numeric, 37
- predicate-trend, 38
- predicate-type, 39
- predicate-utils, 40

- remove_constant, 14, 41, 42–49
- remove_constant, ANY-method
(remove_constant), 41
- remove_constant-method
(remove_constant), 41
- remove_empty, 14, 41, 42, 43–49
- remove_empty, ANY-method (remove_empty),
42
- remove_empty-method (remove_empty), 42
- remove_Inf, 14, 41, 42, 43, 44–49
- remove_Inf, ANY-method (remove_Inf), 43
- remove_Inf-method (remove_Inf), 43
- remove_NA, 14, 41–43, 44, 45–49
- remove_NA, ANY-method (remove_NA), 44
- remove_NA-method (remove_NA), 44
- remove_zero, 14, 41–44, 45, 46–49
- remove_zero, ANY-method (remove_zero), 45
- remove_zero-method (remove_zero), 45
- replace_empty, 14, 41–45, 46, 47–49
- replace_empty, data.frame-method
(replace_empty), 46
- replace_empty, matrix-method
(replace_empty), 46
- replace_empty-method (replace_empty), 46
- replace_Inf, 14, 41–46, 47, 48, 49
- replace_Inf, data.frame-method
(replace_Inf), 47
- replace_Inf, matrix-method
(replace_Inf), 47
- replace_Inf-method (replace_Inf), 47
- replace_NA, 14, 41–47, 48, 49
- replace_NA, data.frame-method
(replace_NA), 48
- replace_NA, matrix-method (replace_NA),
48
- replace_NA-method (replace_NA), 48
- replace_zero, 14, 41–48, 49
- replace_zero, data.frame-method
(replace_zero), 49
- replace_zero, matrix-method
(replace_zero), 49
- replace_zero-method (replace_zero), 49

- scale_range, 50
- seek, 3, 12, 15, 21, 23, 25, 26, 31, 51
- seek_columns (seek), 51
- seek_columns, data.frame-method (seek),
51
- seek_columns-method (seek), 51
- seek_rows (seek), 51
- seek_rows, data.frame-method (seek), 51
- seek_rows-method (seek), 51
- sparsity, 22, 52
- sparsity, data.frame-method (sparsity),
52
- sparsity, matrix-method (sparsity), 52
- sparsity-method (sparsity), 52
- startsWith(), 25, 51
- stats::density(), 27

- trimws(), 14

- validate, 4–11, 53
- vector, 40

- zero (remove_zero), 45