Package: arkhe (via r-universe)

October 31, 2026

Title Tools for Cleaning Rectangular Data

Version 1.11.0.9000

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://codeberg.org/tesselle/arkhe, https://tesselle.r-universe.dev/arkhe, https://packages.tesselle.org/arkhe/

BugReports https://codeberg.org/tesselle/arkhe/issues

Depends R (>= 3.5)

Imports methods, stats, utils

Suggests tinytest

Encoding UTF-8

Roxygen list(markdown = TRUE)

X-schema.org-isPartOf https://www.tesselle.org

X-schema.org-keywords data-cleaning, statistics, r-package

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

Config/roxygen2/version 8.0.0

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

Date/Publication 2026-05-04 14:39:34 UTC

RemoteUrl https://codeberg.org/tesselle/arkhe

RemoteRef HEAD

RemoteSha fb4fa8c98d852c818b514c9545150daaa4e436f2

Contents

append_column . . . 3
append_rownames . . . 4
assert_constant . . . 5
assert_dim . . . 6
assert_empty . . . 7
assert_infinite . . . 7
assert_length . . . 8
assert_lower . . . 9
assert_missing . . . 10
assert_names . . . 10
assert_numeric . . . 11
assert_package . . . 12
assert_square . . . 13
assert_type . . . 14
assert_unique . . . 15
assign . . . 15
bootstrap . . . 16
clean_whitespace . . . 18
compact . . . 19
concat . . . 21
confidence_binomial . . . 21
confidence_bootstrap . . . 23
confidence_mean . . . 24
confidence_multinomial . . . 25
count . . . 27
describe . . . 28
detect . . . 29
discard . . . 30
get . . . 32
interval_credible . . . 33
interval_hdr . . . 34
is_scalar . . . 35
jackknife . . . 36
keep . . . 37
math_gcd . . . 39
math_lcm . . . 39
null . . . 40
predicate-attributes . . . 41
predicate-data . . . 41
predicate-matrix . . . 42
predicate-names . . . 43
predicate-numeric . . . 43
predicate-trend . . . 44
predicate-type . . . 45
remove_constant . . . 46
remove_empty . . . 47

remove_Inf 48
remove_NA 49
remove_zero 50
replace_empty 51
replace_Inf 52
replace_NA 53
replace_zero 54
resample_multinomial 55
resample_uniform 56
scale_midpoint 57
scale_range 58
seek 58
sparsity 60
validate 61

Index **62**

`append_column` *Add a (Named) Vector as a Column*

Description

Add a (Named) Vector as a Column

Usage

```
append_column(x, ...)

## S4 method for signature 'data.frame'
append_column(x, column, after = 0, var = ".col")
```

Arguments

`x`	A `data.frame`.
`...`	Currently not used.
`column`	A (named) `vector`.
`after`	An `integer` specifying a subscript, after which the new column is to be appended.
`var`	A `character` string giving the name of the new column.

Details

If `column` is named, names will be matched to the row names of `x`. Only the first match is retained, and elements of `column` without a match are removed. This allows to add as a column a vector whose length is less than the number of rows in `x` (`NA`s will be inserted).

Value

A data.frame.

Author(s)

N. Frerebeau

See Also

Other data preparation tools: append_rownames(), assign, compact(), count(), detect(), discard(), get, keep(), seek()

Examples

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

Y <- c(D = 44, B = 55, Z = 22)

append_column(X, Y, after = 3)
```

---

append_rownames *Convert Row Names to an Explicit Column*

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 name of the new column.

Value

A data.frame.

Author(s)

N. Frerebeau

See Also

Other data preparation tools: append_column(), 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 *Check Numeric Trend*

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 checking methods: assert_dim(), assert_empty(), assert_infinite(), assert_length(), assert_lower(), assert_missing(), assert_names(), assert_numeric, assert_package(), assert_square(), assert_type(), assert_unique()

`assert_dim` *Check Object Dimensions*

Description

Check Object Dimensions

Usage

```
assert_dim(x, expected)

assert_nrow(x, expected)

assert_ncol(x, expected)
```

Arguments

`x`	An object to be checked.
`expected`	An appropriate expected value.

Value

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

Author(s)

N. Frerebeau

See Also

Other checking methods: assert_constant(), assert_empty(), assert_infinite(), assert_length(), assert_lower(), assert_missing(), assert_names(), assert_numeric, assert_package(), assert_square(), assert_type(), assert_unique()

assert_empty *Check Object Filling*

Description

Checks if an object is (not) empty.

Usage

```
assert_empty(x)

assert_filled(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 checking methods: assert_constant(), assert_dim(), assert_infinite(), assert_length(), assert_lower(), assert_missing(), assert_names(), assert_numeric, assert_package(), assert_square(), assert_type(), assert_unique()

assert_infinite *Check Infinite Values*

Description

Checks if an object contains any infinite (Inf) values.

Usage

```
assert_infinite(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 checking methods: assert_constant(), assert_dim(), assert_empty(), assert_length(), assert_lower(), assert_missing(), assert_names(), assert_numeric, assert_package(), assert_square(), assert_type(), assert_unique()

---

assert_length *Check Object Length(s)*

Description

Check Object Length(s)

Usage

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

assert_lengths(x, expected)
```

Arguments

x	An object to be checked.
expected	An appropriate expected value.
allow_empty	A logical scalar: should empty object be ignored?
allow_null	A logical scalar: should NULL object be ignored?
empty	Deprecated.

Value

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

Author(s)

N. Frerebeau

See Also

Other checking methods: assert_constant(), assert_dim(), assert_empty(), assert_infinite(), assert_lower(), assert_missing(), assert_names(), assert_numeric, assert_package(), assert_square(), assert_type(), assert_unique()

---

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 checking methods: assert_constant(), assert_dim(), assert_empty(), assert_infinite(), assert_length(), assert_missing(), assert_names(), assert_numeric, assert_package(), assert_square(), assert_type(), assert_unique()

assert_missing *Check Missing Values*

Description

Checks if an object contains any missing (NA, NaN) values.

Usage

```
assert_missing(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 checking methods: assert_constant(), assert_dim(), assert_empty(), assert_infinite(), assert_length(), assert_lower(), assert_names(), assert_numeric, assert_package(), assert_square(), assert_type(), assert_unique()

assert_names *Check Object Names*

Description

Check Object Names

Usage

```
assert_names(x, expected = NULL)

assert_rownames(x, expected = NULL)

assert_colnames(x, expected = NULL)
```

Arguments

x	An object to be checked.
expected	An appropriate expected value.

Value

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

Author(s)

N. Frerebeau

See Also

Other checking methods: assert_constant(), assert_dim(), assert_empty(), assert_infinite(), assert_length(), assert_lower(), assert_missing(), assert_numeric, assert_package(), assert_square(), assert_type(), assert_unique()

---

assert_numeric *Check Numeric Values*

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 checking methods: assert_constant(), assert_dim(), assert_empty(), assert_infinite(), assert_length(), assert_lower(), assert_missing(), assert_names(), assert_package(), assert_square(), assert_type(), assert_unique()

---

assert_package *Check the Availability of a Package*

Description

Check the Availability of a Package

Usage

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

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.

Value

Invisibly returns NULL.

Author(s)

N. Frerebeau

See Also

Other checking methods: assert_constant(), assert_dim(), assert_empty(), assert_infinite(), assert_length(), assert_lower(), assert_missing(), assert_names(), assert_numeric, assert_square(), assert_type(), assert_unique()

---

assert_square *Check Matrix*

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 checking methods: assert_constant(), assert_dim(), assert_empty(), assert_infinite(), assert_length(), assert_lower(), assert_missing(), assert_names(), assert_numeric, assert_package(), assert_type(), assert_unique()

assert_type *Check Data Types*

Description

Check Data Types

Usage

```
assert_type(x, expected, allow_empty = TRUE, allow_null = FALSE)

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".
allow_empty	A logical scalar: should empty object be allowed?
allow_null	A logical scalar: should NULL object be ignored?

Value

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

Author(s)

N. Frerebeau

See Also

Other checking methods: assert_constant(), assert_dim(), assert_empty(), assert_infinite(), assert_length(), assert_lower(), assert_missing(), assert_names(), assert_numeric, assert_package(), assert_square(), assert_unique()

assert_unique *Check Duplicates*

Description

Checks if an object contains duplicated elements.

Usage

```
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 checking methods: assert_constant(), assert_dim(), assert_empty(), assert_infinite(), assert_length(), assert_lower(), assert_missing(), assert_names(), assert_numeric, assert_package(), assert_square(), assert_type()

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_column(), append_rownames(), 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 *Nonparametric 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,
  level = 0.95,
  interval = c("basic", "normal", "percentiles")
)
```

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.
level	A length-one numeric vector giving the confidence level. Must be a single number between 0 and 1. Only used if f is NULL.
interval	A character string giving the type of confidence interval to be returned. It must be one "basic" (the default), "normal" or "percentiles" (see confidence_bootstrap()). Any unambiguous substring can be given. Only used if f is NULL.

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.

lower The lower limit of the bootstrap confidence interval at level.

upper The upper limit of the bootstrap confidence interval at level.

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

Author(s)

N. Frerebeau

References

Davison, A. C. & Hinkley, D. V. (1997). *Bootstrap Methods and Their Application*. Cambridge Series on Statistical and Probabilistic Mathematics. Cambridge: Cambridge University Press.

See Also

confidence_bootstrap()

Other resampling methods: jackknife(), resample_multinomial(), resample_uniform()

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 estimates
(z <- bootstrap(x, n = 100, do = mean, f = function(x) { x }))

## Basic bootstrap confidence interval
confidence_bootstrap(z, level = 0.95, type = "basic", t0 = mean(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_columns(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_columns(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_column(), append_rownames(), 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 *Concatenate*

Description

Concatenates character vectors.

Usage

```
x %+% y
```

Arguments

x, y	A character vector.

Value

A character vector.

See Also

Other utilities: null

confidence_binomial *Confidence Interval for Binomial Proportions*

Description

Computes 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_bootstrap(), confidence_mean(), confidence_multinomial(), interval_credible(), interval_hdr()

Examples

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

mean(y)
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_bootstrap *Nonparametric Bootstrap Confidence Interval*

Description

Computes equi-tailed two-sided nonparametric confidence interval.

Usage

```
confidence_bootstrap(object, ...)

## S4 method for signature 'numeric'
confidence_bootstrap(
  object,
  level = 0.95,
  type = c("basic", "normal", "student", "percentiles"),
  t0 = NULL,
  var_t0 = NULL,
  var_t = NULL,
  ...
)
```

Arguments

object	A numeric vector giving the bootstrap replicates of the statistic of interest.
...	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 "basic" (the default), "student", "normal" or "percentiles". Any unambiguous substring can be given.
t0	A length-one numeric vector giving the observed value of the statistic of interest. Must be defined if type is "basic", "student" or "normal".
var_t0	A length-one numeric vector giving an estimate of the variance of the statistic of interest. Must be defined if type is "student". If var_t0 is undefined and type is "normal, it defaults to var(object).
var_t	A numeric vector giving the variances of the bootstrap replicates of the variable of interest. Must be defined if type is "student".

Value

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

Author(s)

N. Frerebeau

References

Davison, A. C. & Hinkley, D. V. (1997). *Bootstrap Methods and Their Application*. Cambridge Series on Statistical and Probabilistic Mathematics. Cambridge: Cambridge University Press.

See Also

bootstrap()

Other summary statistics: confidence_binomial(), confidence_mean(), confidence_multinomial(), interval_credible(), interval_hdr()

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 estimates
(z <- bootstrap(x, n = 100, do = mean, f = function(x) { x }))

## Basic bootstrap confidence interval
confidence_bootstrap(z, level = 0.95, type = "basic", t0 = mean(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_bootstrap(), confidence_multinomial(), interval_credible(), interval_hdr()

Examples

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

mean(y)
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 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

object	A numeric vector of positive integers giving the number of occurrences of each class.
...	Currently not used.
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 two column numeric matrix giving the lower and upper confidence limits.

Author(s)

N. Frerebeau

See Also

Other summary statistics: confidence_binomial(), confidence_bootstrap(), confidence_mean(), interval_credible(), interval_hdr()

Examples

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

mean(y)
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_column(), append_rownames(), assign, compact(), detect(), discard(), get, keep(), seek()

Examples

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

## Add NA
k <- sample(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(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

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?

Value

A logical vector.

Author(s)

N. Frerebeau

See Also

Other data preparation tools: append_column(), append_rownames(), assign, compact(), count(), discard(), get, keep(), seek()

Examples

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

## Add NA
k <- sample(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_columns(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_columns(
  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_column(), append_rownames(), assign, compact(), count(), detect(), get, keep(), seek()

Examples

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

## Add NA
k <- sample(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 'ANY'
get_columns(x, select = NULL, names = NULL, ...)

## S4 method for signature 'ANY'
get_rows(x, select = NULL, names = 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()) that returns an integer or logical vector.
names	A character vector of row/column names to look for. Only used if select is NULL.

Value

An object of the same sort as x.

Author(s)

N. Frerebeau

See Also

Other data preparation tools: append_column(), append_rownames(), assign, compact(), count(), detect(), discard(), keep(), seek()

Examples

```
## Seek columns
seek_columns(iris, select = startsWith, prefix = "Petal")
seek_columns(iris, names = c("Petal.Length", "Petal.Width"))

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

x <- get_columns(iris, names = c("Petal.Length", "Petal.Width"))
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_bootstrap()`, `confidence_mean()`, `confidence_multinomial()`, `interval_hdr()`

Examples

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

---

interval_hdr *Highest Density Regions*

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_bootstrap()`, `confidence_mean()`, `confidence_multinomial()`, `interval_credible()`

Examples

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

---

is_scalar *Scalar Type Predicates*

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-attributes, predicate-data, predicate-matrix, predicate-names, predicate-numeric, predicate-trend, predicate-type

---

jackknife *Jackknife Estimation*

Description

Jackknife Estimation

Usage

```
jackknife(object, ...)

## S4 method for signature 'numeric'
jackknife(object, do, ..., 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.
f	A function that takes a single numeric vector (the leave-one-out values of do) 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 Also

Other resampling methods: bootstrap(), resample_multinomial(), resample_uniform()

Examples

```
x <- rnorm(20)

## 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 *Keep Rows/Columns Using a Predicate*

Description

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

Usage

```
keep(x, ...)

keep_columns(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_columns(
  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_column(), append_rownames(), assign, compact(), count(), detect(), discard(), get, seek()

Examples

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

## Add NA
k <- sample(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 *Least Common Multiple*

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 *Default value for NULL*

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

predicate-attributes *Attributes Predicates*

Description

- `has_length()` checks how long is an object.
- `is_empty()` checks is an object is empty (any zero-length dimensions).

Usage

```
has_length(x, n = NULL)

is_empty(x)
```

Arguments

x	A vector to be tested.
n	A length-one numeric vector specifying the length to test x with. If NULL, returns TRUE if x has length greater than zero, and FALSE otherwise.

Value

A logical scalar.

See Also

Other predicates: is_scalar, predicate-data, predicate-matrix, predicate-names, predicate-numeric, predicate-trend, predicate-type

predicate-data *Utility Predicates*

Description

- `has_missing()` and `has_infinite()` check if an object contains missing or infinite values.
- `has_duplicates()` checks if an object has duplicated elements.

Usage

```
has_missing(x)

has_infinite(x)

has_duplicates(x)

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

Arguments

x	A vector to be tested.
tolerance	A numeric scalar giving the tolerance to check within (for numeric vector).
na.rm	A logical scalar: should missing values (including NaN) be omitted?

Value

A logical scalar.

See Also

Other predicates: is_scalar, predicate-attributes, predicate-matrix, predicate-names, predicate-numeric, predicate-trend, predicate-type

---

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-attributes, predicate-data, predicate-names, predicate-numeric, predicate-trend, predicate-type

predicate-names *Names Predicates*

Description

Checks if an object is named.

Usage

```
has_names(x, names = NULL)

has_rownames(x, names = NULL)

has_colnames(x, names = NULL)
```

Arguments

x	A vector to be tested.
names	A character vector specifying the names to test x with. If NULL, returns TRUE if x has names, and FALSE otherwise.

Value

A logical scalar.

See Also

Other predicates: is_scalar, predicate-attributes, predicate-data, predicate-matrix, predicate-numeric, predicate-trend, predicate-type

predicate-numeric *Numeric Predicates*

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

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

Value

A logical vector.

See Also

Other predicates: is_scalar, predicate-attributes, predicate-data, predicate-matrix, predicate-names, predicate-trend, predicate-type

---

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

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

Value

A logical scalar.

See Also

Other predicates: is_scalar, predicate-attributes, predicate-data, predicate-matrix, predicate-names, predicate-numeric, predicate-type

---

predicate-type *Type Predicates*

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-attributes, predicate-data, predicate-matrix, predicate-names, predicate-numeric, predicate-trend

---

remove_constant *Remove Constant Columns*

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 *Remove Rows/Columns with Empty String*

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(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 *Remove Rows/Columns with Infinite Values*

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(10, 25, TRUE), nrow = 5, ncol = 5)

## Add Inf
k <- sample(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(10, 25, TRUE), nrow = 5, ncol = 5)

## Add NA
k <- sample(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 *Remove Rows/Columns with Zeros*

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(10, 25, TRUE), nrow = 5, ncol = 5)

## Add zero
k <- sample(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 *Replace Empty String*

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(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` *Replace Infinite Values*

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(10, 25, TRUE), nrow = 5, ncol = 5)

## Add Inf
k <- sample(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 *Replace Missing Values*

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(10, 25, TRUE), nrow = 5, ncol = 5)

## Add NA
k <- sample(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 *Replace Zeros*

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(10, 25, TRUE), nrow = 5, ncol = 5)

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

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

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

`resample_multinomial` *Draw Multinomial Random Sample*

Description

Draws a random (sub)sample from a multinomial distribution.

Usage

```
resample_multinomial(object, ...)

## S4 method for signature 'numeric'
resample_multinomial(object, n, size = sum(object), ...)
```

Arguments

`object`	A length-k `integer` vector, specifying the probability for the k classes; is internally normalized to sum to 1.
`...`	Currently not used.
`n`	A non-negative `integer` specifying the number of random vector to draw.
`size`	A non-negative `integer` specifying the sample size.

Value

A numeric matrix with n rows and k columns.

Author(s)

N. Frerebeau

See Also

stats::rmultinom()

Other resampling methods: bootstrap(), jackknife(), resample_uniform()

Examples

```
## Uniform distribution
x <- rnorm(20)
resample_uniform(x, n = 10)

## Multinomial distribution
x <- sample(100, 20, TRUE)
resample_multinomial(x, n = 10)
```

---

resample_uniform *Draw Uniform Random Sample*

Description

Draws a random (sub)sample (with or without replacement).

Usage

```
resample_uniform(object, ...)

## S4 method for signature 'numeric'
resample_uniform(object, n, size = length(object), replace = FALSE, ...)
```

Arguments

object	A numeric vector.
...	Currently not used.
n	A non-negative integer specifying the number of random vector to draw.
size	A non-negative integer specifying the sample size.
replace	A logical scalar: should sampling be with replacement?

Value

A numeric matrix with n rows and size columns.

Author(s)

N. Frerebeau

See Also

Other resampling methods: `bootstrap()`, `jackknife()`, `resample_multinomial()`

Examples

```
## Uniform distribution
x <- rnorm(20)
resample_uniform(x, n = 10)

## Multinomial distribution
x <- sample(100, 20, TRUE)
resample_multinomial(x, n = 10)
```

---

scale_midpoint *Rescale Continuous Vector (minimum, midpoint, maximum)*

Description

Rescales continuous vector to have specified minimum, midpoint and maximum.

Usage

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

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.
`midpoint`	A length-one `numeric` vector specifying the mid-point of input range.

Value

A `numeric` vector.

Note

For internal use only.

See Also

Other scales: `scale_range()`

scale_range *Rescale Continuous Vector (minimum, maximum)*

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.

See Also

Other scales: scale_midpoint()

seek *Search Rows/Columns by Name*

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, names = NULL, ...)
```

```
## S4 method for signature 'matrix'
seek_rows(x, select = NULL, names = NULL, ...)

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

## S4 method for signature 'matrix'
seek_columns(x, select = NULL, names = 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()) that returns an integer or logical vector.
names	A character vector of row/column names to look for. Only used if select is NULL.

Value

An integer vector or NULL.

Author(s)

N. Frerebeau

See Also

Other data preparation tools: append_column(), append_rownames(), assign, compact(), count(), detect(), discard(), get, keep()

Examples

```
## Seek columns
seek_columns(iris, select = startsWith, prefix = "Petal")
seek_columns(iris, names = c("Petal.Length", "Petal.Width"))

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

x <- get_columns(iris, names = c("Petal.Length", "Petal.Width"))
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

x	An R object (should be a matrix or a data.frame).
...	Currently not used.
count	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(15, 3, FALSE)
X[k] <- NA
```

```
## Sparsity
sparsity(X)

## Quick description
describe(X)
```

`validate` *Validate a Condition*

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

Index

∗ **checking methods**
  assert_constant, 5
  assert_dim, 6
  assert_empty, 7
  assert_infinite, 7
  assert_length, 8
  assert_lower, 9
  assert_missing, 10
  assert_names, 10
  assert_numeric, 11
  assert_package, 12
  assert_square, 13
  assert_type, 14
  assert_unique, 15
∗ **data cleaning tools**
  clean_whitespace, 18
  remove_constant, 46
  remove_empty, 47
  remove_Inf, 48
  remove_NA, 49
  remove_zero, 50
  replace_empty, 51
  replace_Inf, 52
  replace_NA, 53
  replace_zero, 54
∗ **data preparation tools**
  append_column, 3
  append_rownames, 4
  assign, 15
  compact, 19
  count, 27
  detect, 29
  discard, 30
  get, 32
  keep, 37
  seek, 58
∗ **data summaries**
  describe, 28
  sparsity, 60
∗ **mathematic functions**
  math_gcd, 39
  math_lcm, 39
∗ **predicates**
  is_scalar, 35
  predicate-attributes, 41
  predicate-data, 41
  predicate-matrix, 42
  predicate-names, 43
  predicate-numeric, 43
  predicate-trend, 44
  predicate-type, 45
∗ **resampling methods**
  bootstrap, 16
  jackknife, 36
  resample_multinomial, 55
  resample_uniform, 56
∗ **scales**
  scale_midpoint, 57
  scale_range, 58
∗ **summary statistics**
  confidence_binomial, 21
  confidence_bootstrap, 23
  confidence_mean, 24
  confidence_multinomial, 25
  interval_credible, 33
  interval_hdr, 34
∗ **utilities**
  concat, 21
  null, 40
∗ **validation methods**
  validate, 61
%+% *(*concat*)*, 21

append_column, 3
append_column(), *5*, *16*, *20*, *27*, *30, 31*, *33*, *38*, *59*
append_column,data.frame-method *(*append_column*)*, 3
append_column-method *(*append_column*)*, 3

append_rownames, 4
append_rownames(), *4*, *16*, *20*, *27*, *30*, *31*, *33*, *38*, *59*
append_rownames,data.frame-method (append_rownames), 4
append_rownames-method (append_rownames), 4
assert_colnames (assert_names), 10
assert_constant, 5
assert_constant(), *6–15*
assert_count (assert_numeric), 11
assert_decreasing (assert_constant), 5
assert_dim, 6
assert_dim(), *6–15*
assert_empty, 7
assert_empty(), *6*, *8–15*
assert_even (assert_numeric), 11
assert_filled (assert_empty), 7
assert_function (assert_type), 14
assert_greater (assert_lower), 9
assert_increasing (assert_constant), 5
assert_infinite, 7
assert_infinite(), *6*, *7*, *9–15*
assert_length, 8
assert_length(), *6–15*
assert_lengths (assert_length), 8
assert_lower, 9
assert_lower(), *6–15*
assert_missing, 10
assert_missing(), *6–9*, *11–15*
assert_names, 10
assert_names(), *6–10*, *12–15*
assert_ncol (assert_dim), 6
assert_negative (assert_numeric), 11
assert_nrow (assert_dim), 6
assert_numeric, *6–10*, 11, *11*, *13–15*
assert_odd (assert_numeric), 11
assert_package, 12
assert_package(), *6–15*
assert_positive (assert_numeric), 11
assert_rownames (assert_names), 10
assert_scalar (assert_type), 14
assert_square, 13
assert_square(), *6–15*
assert_symmetric (assert_square), 13
assert_type, 14
assert_type(), *6–13*, *15*
assert_unique, 15
assert_unique(), *6–14*
assert_whole (assert_numeric), 11
assign, *4*, *5*, 15, *20*, *27*, *30*, *31*, *33*, *38*, *59*
assign_colnames (assign), 15
assign_colnames,data.frame-method (assign), 15
assign_colnames-method (assign), 15
assign_rownames (assign), 15
assign_rownames,data.frame-method (assign), 15
assign_rownames-method (assign), 15

bootstrap, 16
bootstrap(), *24*, *36*, *56*, *57*
bootstrap,numeric-method (bootstrap), 16
bootstrap-method (bootstrap), 16

character, *3*, *4*, *12*, *14*, *17*, *19*, *21–23*, *25*, *26*, *32*, *43*, *59*
clean_whitespace, 18
clean_whitespace(), *47–55*
clean_whitespace,data.frame-method (clean_whitespace), 18
clean_whitespace,matrix-method (clean_whitespace), 18
clean_whitespace-method (clean_whitespace), 18
compact, 19
compact(), *4*, *5*, *16*, *27*, *30*, *31*, *33*, *38*, *59*
compact,ANY-method (compact), 19
compact-method (compact), 19
compact_columns (compact), 19
compact_columns,ANY-method (compact), 19
compact_columns-method (compact), 19
compact_rows (compact), 19
compact_rows,ANY-method (compact), 19
compact_rows-method (compact), 19
concat, 21, *40*
confidence_binomial, 21
confidence_binomial(), *24–26*, *34*, *35*
confidence_binomial,numeric-method (confidence_binomial), 21
confidence_binomial-method (confidence_binomial), 21
confidence_bootstrap, 23
confidence_bootstrap(), *17*, *18*, *22*, *25*, *26*, *34*, *35*
confidence_bootstrap,numeric-method (confidence_bootstrap), 23

confidence_bootstrap-method (confidence_bootstrap), 23
confidence_mean, 24
confidence_mean(), 22, 24, 26, 34, 35
confidence_mean,numeric-method (confidence_mean), 24
confidence_mean-method (confidence_mean), 24
confidence_multinomial, 25
confidence_multinomial(), 22, 24, 25, 34, 35
confidence_multinomial,numeric-method (confidence_multinomial), 25
confidence_multinomial-method (confidence_multinomial), 25
count, 27
count(), 4, 5, 16, 20, 30, 31, 33, 38, 59
count,data.frame-method (count), 27
count,matrix-method (count), 27
count-method (count), 27

data.frame, 3–5, 16, 19, 20, 27–29, 31, 32, 38, 46–54, 59, 60
describe, 28
describe(), 60
describe,ANY-method (describe), 28
describe-method (describe), 28
detect, 29
detect(), 4, 5, 16, 20, 27, 31, 33, 38, 59
detect,ANY-method (detect), 29
detect-method (detect), 29
discard, 30
discard(), 4, 5, 16, 20, 27, 30, 33, 38, 59
discard,ANY-method (discard), 30
discard-method (discard), 30
discard_columns (discard), 30
discard_columns,ANY-method (discard), 30
discard_columns-method (discard), 30
discard_rows (discard), 30
discard_rows,ANY-method (discard), 30
discard_rows-method (discard), 30

empty, 8, 14

function, 17, 27, 29, 31, 32, 36, 38, 59

get, 4, 5, 16, 20, 27, 30, 31, 32, 38, 59
get_columns (get), 32
get_columns,ANY-method (get), 32
get_columns-method (get), 32
get_rows (get), 32
get_rows,ANY-method (get), 32
get_rows-method (get), 32

has_colnames (predicate-names), 43
has_duplicates (predicate-data), 41
has_infinite (predicate-data), 41
has_length (predicate-attributes), 41
has_missing (predicate-data), 41
has_names (predicate-names), 43
has_rownames (predicate-names), 43

infinite values, 48, 52
integer, 3, 17, 55, 56, 59
interval_credible, 33
interval_credible(), 22, 24–26, 35
interval_credible,numeric-method (interval_credible), 33
interval_credible-method (interval_credible), 33
interval_hdr, 34
interval_hdr(), 22, 24–26, 34
interval_hdr,numeric,missing-method (interval_hdr), 34
interval_hdr,numeric,numeric-method (interval_hdr), 34
interval_hdr-method (interval_hdr), 34
is_atomic (predicate-type), 45
is_character (predicate-type), 45
is_constant (predicate-trend), 44
is_decreasing (predicate-trend), 44
is_double (predicate-type), 45
is_empty (predicate-attributes), 41
is_error (predicate-type), 45
is_even (predicate-numeric), 43
is_greater (predicate-trend), 44
is_increasing (predicate-trend), 44
is_integer (predicate-type), 45
is_list (predicate-type), 45
is_logical (predicate-type), 45
is_lower (predicate-trend), 44
is_message (predicate-type), 45
is_negative (predicate-numeric), 43
is_numeric (predicate-type), 45
is_odd (predicate-numeric), 43
is_positive (predicate-numeric), 43
is_scalar, 35, 41–46
is_scalar_atomic (is_scalar), 35

is_scalar_character (is_scalar), 35
is_scalar_double (is_scalar), 35
is_scalar_integer (is_scalar), 35
is_scalar_list (is_scalar), 35
is_scalar_logical (is_scalar), 35
is_scalar_numeric (is_scalar), 35
is_scalar_vector (is_scalar), 35
is_square (predicate-matrix), 42
is_symmetric (predicate-matrix), 42
is_unique (predicate-data), 41
is_vector (predicate-type), 45
is_warning (predicate-type), 45
is_whole (predicate-numeric), 43
is_zero (predicate-numeric), 43

jackknife, 36
jackknife(), *18*, *56, 57*
jackknife,numeric-method (jackknife), 36
jackknife-method (jackknife), 36

keep, 37
keep(), *4, 5*, *16*, *20*, *27*, *30, 31*, *33*, *59*
keep,ANY-method (keep), 37
keep-method (keep), 37
keep_columns (keep), 37
keep_columns,ANY-method (keep), 37
keep_columns-method (keep), 37
keep_rows (keep), 37
keep_rows,ANY-method (keep), 37
keep_rows-method (keep), 37

logical, *4*, *8*, *11, 12*, *14*, *16*, *19, 20*, *22*, *26, 27*, *29*, *31*, *35*, *38*, *41–50*, *56*, *60*

math_gcd, 39
math_gcd(), *40*
math_gcd,numeric,numeric-method (math_gcd), 39
math_gcd-method (math_gcd), 39
math_lcm, 39
math_lcm(), *39*
math_lcm,numeric,numeric-method (math_lcm), 39
math_lcm-method (math_lcm), 39
matrix, *13*, *19, 20*, *27–29*, *31–34*, *38*, *42*, *46–54*, *56*, *59, 60*
missing (remove_NA), 49
missing values, *49*, *53*

null, *21*, 40

numeric, *4, 5*, *9*, *11*, *16, 17*, *20*, *22–27*, *29*, *31*, *33, 34*, *36*, *38–42*, *44, 45*, *47–50*, *56–58*, *60*

predicate-attributes, 41
predicate-data, 41
predicate-matrix, 42
predicate-names, 43
predicate-numeric, 43
predicate-trend, 44
predicate-type, 45

remove_constant, 46
remove_constant(), *19*, *48–55*
remove_constant,ANY-method (remove_constant), 46
remove_constant-method (remove_constant), 46
remove_empty, 47
remove_empty(), *19*, *47*, *49–55*
remove_empty,ANY-method (remove_empty), 47
remove_empty-method (remove_empty), 47
remove_Inf, 48
remove_Inf(), *19*, *47, 48*, *50–55*
remove_Inf,ANY-method (remove_Inf), 48
remove_Inf-method (remove_Inf), 48
remove_NA, 49
remove_NA(), *19*, *47–49*, *51–55*
remove_NA,ANY-method (remove_NA), 49
remove_NA-method (remove_NA), 49
remove_zero, 50
remove_zero(), *19*, *47–50*, *52–55*
remove_zero,ANY-method (remove_zero), 50
remove_zero-method (remove_zero), 50
replace_empty, 51
replace_empty(), *19*, *47–51*, *53–55*
replace_empty,data.frame-method (replace_empty), 51
replace_empty,matrix-method (replace_empty), 51
replace_empty-method (replace_empty), 51
replace_Inf, 52
replace_Inf(), *19*, *47–52*, *54, 55*
replace_Inf,data.frame-method (replace_Inf), 52
replace_Inf,matrix-method (replace_Inf), 52
replace_Inf-method (replace_Inf), 52

replace_NA, 53
replace_NA(), *19*, *47–53*, *55*
replace_NA,data.frame-method *(replace_NA)*, 53
replace_NA,matrix-method *(replace_NA)*, 53
replace_NA-method *(replace_NA)*, 53
replace_zero, 54
replace_zero(), *19*, *47–54*
replace_zero,data.frame-method *(replace_zero)*, 54
replace_zero,matrix-method *(replace_zero)*, 54
replace_zero-method *(replace_zero)*, 54
resample_multinomial, 55
resample_multinomial(), *18*, *36*, *57*
resample_multinomial,numeric-method *(resample_multinomial)*, 55
resample_multinomial-method *(resample_multinomial)*, 55
resample_uniform, 56
resample_uniform(), *18*, *36*, *56*
resample_uniform,numeric-method *(resample_uniform)*, 56
resample_uniform-method *(resample_uniform)*, 56

scale_midpoint, 57
scale_midpoint(), *58*
scale_range, 58
scale_range(), *57*
seek, 58
seek(), *4, 5*, *16*, *20*, *27*, *30, 31*, *33*, *38*
seek_columns *(seek)*, 58
seek_columns,data.frame-method *(seek)*, 58
seek_columns,matrix-method *(seek)*, 58
seek_columns-method *(seek)*, 58
seek_rows *(seek)*, 58
seek_rows,data.frame-method *(seek)*, 58
seek_rows,matrix-method *(seek)*, 58
seek_rows-method *(seek)*, 58
sparsity, 60
sparsity(), *28*
sparsity,data.frame-method *(sparsity)*, 60
sparsity,matrix-method *(sparsity)*, 60
sparsity-method *(sparsity)*, 60
startsWith(), *32*, *59*
stats::density(), *34*
stats::rmultinom(), *56*

trimws(), *19*

validate, 61
vector, *41–43*

zero *(remove_zero)*, 50