

# Package ‘arkhe’

May 12, 2025

**Title** Tools for Cleaning Rectangular Data

**Version** 1.11.0

**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://packages.tesselle.org/arkhe/>

**BugReports** <https://codeberg.org/tesselle/arkhe/issues>

**Depends** R (>= 3.5)

**Imports** methods, stats, utils

**Suggests** tinytest

**Encoding** UTF-8

**RoxygenNote** 7.3.2.9000

**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'

**NeedsCompilation** no

**Author** Nicolas Frerebeau [aut, cre] (ORCID:

<<https://orcid.org/0000-0001-5759-4944>>),

Brice Lebrun [art] (ORCID: <<https://orcid.org/0000-0001-7503-8685>>,  
Logo designer),

Université Bordeaux Montaigne [fnd] (ROR: <<https://ror.org/03pbgwk21>>),

CNRS [fnd] (ROR: <<https://ror.org/02feahw73>>)

**Maintainer** Nicolas Frerebeau <nicolas.frerebeau@u-bordeaux-montaigne.fr>

**Repository** CRAN

**Date/Publication** 2025-05-12 17:30:02 UTC

## Contents

append_column . . . . .	3
append_rownames . . . . .	4
assert_constant . . . . .	5
assert_dim . . . . .	6
assert_empty . . . . .	7
assert_infinite . . . . .	8
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	<i>Add a (Named) Vector as a Column</i>
---------------	---

---

**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 <a href="#">data.frame</a> .
...	Currently not used.
column	A (named) vector.
after	An <a href="#">integer</a> specifying a subscript, after which the new column is to be appended.
var	A <a href="#">character</a> 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 (NAs 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 <a href="#">data.frame</a> .
...	Currently not used.
after	A length-one <a href="#">numeric</a> vector specifying a subscript, after which the row names are to be appended.
remove	A <a href="#">logical</a> scalar: should the row names be removed?
var	A <a href="#">character</a> 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 <a href="#">logical</a> scalar: should <a href="#">empty</a> object be ignored?
allow_null	A <a href="#">logical</a> 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 <a href="#">numeric</a> 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	<i>Check Missing Values</i>
----------------	-----------------------------

---

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

---

**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**

<code>x</code>	A <a href="#">character</a> vector naming the packages to check.
<code>ask</code>	A <a href="#">logical</a> 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	<i>Check Data Types</i>
-------------	-------------------------

---

### 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 <a href="#">character</a> string specifying the expected type. It must be one of "list", "atomic", "vector", "numeric", "integer", "double", "character" or "logical".
allow_empty	A <a href="#">logical</a> scalar: should <a href="#">empty</a> object be allowed?
allow_null	A <a href="#">logical</a> 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	<i>Check Duplicates</i>
---------------	-------------------------

---

**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	<i>Assign a Specific Row/Column to the Column/Row Names</i>
--------	---

---

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

**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 <b>numeric</b> vector.
...	Extra arguments to be passed to do.
do	A <b>function</b> that takes object as an argument and returns a single numeric value.
n	A non-negative <b>integer</b> giving the number of bootstrap replications.
f	A <b>function</b> that takes a single numeric vector (the result of do) as argument.
level	A length-one <b>numeric</b> vector giving the confidence level. Must be a single number between 0 and 1. Only used if f is NULL.
interval	A <b>character</b> string giving the type of confidence interval to be returned. It must be one "basic" (the default), "normal" or "percentiles" (see <a href="#">confidence_bootstrap()</a> ). 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	<i>Remove Leading/Trailing Whitespace</i>
------------------	---

---

**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**

<code>x</code>	An R object (should be a <a href="#">matrix</a> or a <a href="#">data.frame</a> ).
<code>...</code>	Currently not used.
<code>which</code>	A <a href="#">character</a> string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right").
<code>squish</code>	A <a href="#">logical</a> 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 <a href="#">matrix</a> or a <a href="#">data.frame</a> ).
...	Currently not used.
margin	A length-one <a href="#">numeric</a> vector giving the subscripts which the function will be applied over (1 indicates rows, 2 indicates columns).
na.rm	A <a href="#">logical</a> scalar: should NA values be stripped before the computation proceeds?
verbose	A <a href="#">logical</a> 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	<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 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 <b>numeric</b> vector giving the number of success.
...	Currently not used.
n	A length-one <b>numeric</b> vector giving the number of trials.
level	A length-one <b>numeric</b> vector giving the confidence level. Must be a single number between 0 and 1.
method	A <b>character</b> string specifying the method to be used. Any unambiguous substring can be used.
corrected	A <b>logical</b> 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 <b>numeric</b> vector giving the bootstrap replicates of the statistic of interest.
...	Currently not used.
level	A length-one <b>numeric</b> vector giving the confidence level. Must be a single number between 0 and 1.
type	A <b>character</b> 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 <b>numeric</b> 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 <b>numeric</b> 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 <b>numeric</b> 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\\_credibile\(\)](#), [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 <a href="#">numeric</a> vector.
...	Currently not used.
level	A length-one <a href="#">numeric</a> 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\\_credibile\(\)](#), [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 <a href="#">numeric</a> vector of positive integers giving the number of occurrences of each class.
...	Currently not used.
level	A length-one <a href="#">numeric</a> vector giving the confidence level. Must be a single number between 0 and 1.
method	A <a href="#">character</a> string specifying the method to be used. Any unambiguous substring can be used.
corrected	A <a href="#">logical</a> 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	<i>Count Values Using a Predicate</i>
-------	---------------------------------------

---

**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 <a href="#">matrix</a> or a <a href="#">data.frame</a> ).
...	Further arguments to be passed to f.
f	A predicate <a href="#">function</a> .
margin	A length-one <a href="#">numeric</a> vector giving the subscripts which the function will be applied over (1 indicates rows, 2 indicates columns).
negate	A <a href="#">logical</a> scalar: should the negation of f be used instead of f?
na.rm	A <a href="#">logical</a> 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(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	<i>Find Rows/Columns Using a Predicate</i>
--------	--

---

**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 <a href="#">matrix</a> or a <a href="#">data.frame</a> ).
<code>...</code>	Further arguments to be passed to <code>f</code> .
<code>f</code>	A predicate <a href="#">function</a> .
<code>margin</code>	A length-one <a href="#">numeric</a> vector giving the subscripts which the function will be applied over (1 indicates rows, 2 indicates columns).
<code>negate</code>	A <a href="#">logical</a> scalar: should the negation of <code>f</code> be used instead of <code>f</code> ?
<code>all</code>	A <a href="#">logical</a> 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 <a href="#">logical</a> 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(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_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

<code>x</code>	An R object (should be a <a href="#">matrix</a> or a <a href="#">data.frame</a> ).
<code>...</code>	Further arguments to be passed to <code>f</code> .
<code>f</code>	A predicate <a href="#">function</a> .
<code>margin</code>	A length-one <a href="#">numeric</a> vector giving the subscripts which the function will be applied over (1 indicates rows, 2 indicates columns).
<code>negate</code>	A <a href="#">logical</a> scalar: should the negation of <code>f</code> be used instead of <code>f</code> ?
<code>all</code>	A <a href="#">logical</a> 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 <a href="#">logical</a> scalar: should NA values be stripped before the computation proceeds?
<code>verbose</code>	A <a href="#">logical</a> 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(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 '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 <a href="#">matrix</a> or a <a href="#">data.frame</a> ).
...	Further arguments to be passed to select.
select	A <a href="#">function</a> to be applied to the row/column names (e.g. <a href="#">startsWith()</a> ) that returns an integer or logical vector.
names	A <a href="#">character</a> 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 <b>numeric</b> vector.
...	Currently not used.
level	A length-one <b>numeric</b> 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	<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 <a href="#">numeric</a> vector giving the coordinates of the points where the density is estimated.
y	A <a href="#">numeric</a> 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 <a href="#">stats::density()</a> .
level	A length-one <a href="#">numeric</a> 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	<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-attributes](#), [predicate-data](#), [predicate-matrix](#), [predicate-names](#), [predicate-numeric](#), [predicate-trend](#), [predicate-type](#)

---

jackknife	<i>Jackknife Estimation</i>
-----------	-----------------------------

---

**Description**

Jackknife Estimation

**Usage**

```
jackknife(object, ...)

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

**Arguments**

object	A <a href="#">numeric</a> vector.
...	Extra arguments to be passed to do.
do	A <a href="#">function</a> that takes object as an argument and returns a single numeric value.
f	A <a href="#">function</a> 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

<code>x</code>	An R object (should be a <a href="#">matrix</a> or a <a href="#">data.frame</a> ).
<code>...</code>	Further arguments to be passed to <code>f</code> .
<code>f</code>	A predicate <a href="#">function</a> .
<code>margin</code>	A length-one <a href="#">numeric</a> vector giving the subscripts which the function will be applied over (1 indicates rows, 2 indicates columns).
<code>negate</code>	A <a href="#">logical</a> scalar: should the negation of <code>f</code> be used instead of <code>f</code> ?
<code>all</code>	A <a href="#">logical</a> 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 <a href="#">logical</a> scalar: should NA values be stripped before the computation proceeds?
<code>verbose</code>	A <a href="#">logical</a> 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(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	<i>Greatest Common Divisor</i>
----------	--------------------------------

---

**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

*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 <a href="#">vector</a> to be tested.
tolerance	A <a href="#">numeric</a> scalar giving the tolerance to check within (for numeric vector).
na.rm	A <a href="#">logical</a> 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 <a href="#">matrix</a> 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 <a href="#">numeric</a> object to be tested.
tolerance	A <a href="#">numeric</a> scalar giving the tolerance to check within.
...	Currently not used.
strict	A <a href="#">logical</a> 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 <a href="#">numeric</a> object to be tested.
tolerance	A <a href="#">numeric</a> scalar giving the tolerance to check within.
na.rm	A <a href="#">logical</a> scalar: should missing values (including NaN) be omitted?
strict	A <a href="#">logical</a> 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	<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-attributes](#), [predicate-data](#), [predicate-matrix](#), [predicate-names](#), [predicate-numeric](#), [predicate-trend](#)

---

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

*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 <a href="#">matrix</a> or a <a href="#">data.frame</a> ).
...	Currently not used.
margin	A length-one <a href="#">numeric</a> vector giving the subscripts which the function will be applied over (1 indicates rows, 2 indicates columns).
all	A <a href="#">logical</a> 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 <a href="#">logical</a> 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

*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 <a href="#">matrix</a> or a <a href="#">data.frame</a> ).
...	Currently not used.
margin	A length-one <a href="#">numeric</a> vector giving the subscripts which the function will be applied over (1 indicates rows, 2 indicates columns).
all	A <a href="#">logical</a> 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 <a href="#">logical</a> 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 <a href="#">matrix</a> or a <a href="#">data.frame</a> ).
...	Currently not used.
margin	A length-one <a href="#">numeric</a> vector giving the subscripts which the function will be applied over (1 indicates rows, 2 indicates columns).
all	A <a href="#">logical</a> 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 <a href="#">logical</a> 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`*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**

<code>x</code>	An R object (should be a <a href="#">matrix</a> or a <a href="#">data.frame</a> ).
<code>...</code>	Currently not used.
<code>margin</code>	A length-one <a href="#">numeric</a> vector giving the subscripts which the function will be applied over (1 indicates rows, 2 indicates columns).
<code>all</code>	A <a href="#">logical</a> 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>verbose</code>	A <a href="#">logical</a> 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 <a href="#">matrix</a> or a <a href="#">data.frame</a> ).
...	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`*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**

<code>x</code>	An R object (should be a <a href="#">matrix</a> or a <a href="#">data.frame</a> ).
<code>...</code>	Currently not used.
<code>value</code>	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

*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 <a href="#">matrix</a> or a <a href="#">data.frame</a> ).
...	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`*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**

<code>x</code>	An R object (should be a <a href="#">matrix</a> or a <a href="#">data.frame</a> ).
<code>...</code>	Currently not used.
<code>value</code>	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)
```

---

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- $k$ <a href="#">integer</a> vector, specifying the probability for the $k$ classes; is internally normalized to sum to 1.
...	Currently not used.
n	A non-negative <a href="#">integer</a> specifying the number of random vector to draw.
size	A non-negative <a href="#">integer</a> 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(1:100, 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 <a href="#">numeric</a> vector.
...	Currently not used.
n	A non-negative <a href="#">integer</a> specifying the number of random vector to draw.
size	A non-negative <a href="#">integer</a> specifying the sample size.
replace	A <a href="#">logical</a> 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(1:100, 20, TRUE)
resample_multinomial(x, n = 10)
```

---

scale_midpoint	<i>Rescale Continuous Vector (minimum, midpoint, maximum)</i>
----------------	---

---

**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 <a href="#">numeric</a> vector.
to	A length-two <a href="#">numeric</a> vector specifying the output range.
from	A length-two <a href="#">numeric</a> vector specifying the input range.
midpoint	A length-one <a href="#">numeric</a> 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	<i>Rescale Continuous Vector (minimum, maximum)</i>
-------------	---

---

**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 <a href="#">numeric</a> vector.
to	A length-two <a href="#">numeric</a> vector specifying the output range.
from	A length-two <a href="#">numeric</a> vector specifying the input range.

**Value**

A [numeric](#) vector.

**Note**

For internal use only.

**See Also**

Other scales: [scale\\_midpoint\(\)](#)

---

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, 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 <a href="#">matrix</a> or a <a href="#">data.frame</a> ).
...	Further arguments to be passed to <code>select</code> .
select	A <a href="#">function</a> to be applied to the row/column names (e.g. <a href="#">startsWith()</a> ) that returns an integer or logical vector.
names	A <a href="#">character</a> vector of row/column names to look for. Only used if <code>select</code> is <code>NULL</code> .

### 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

<code>x</code>	An R object (should be a <a href="#">matrix</a> or a <a href="#">data.frame</a> ).
<code>...</code>	Currently not used.
<code>count</code>	A <a href="#">logical</a> 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

*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, 8
  - 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, 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](#), [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](#), [7–15](#)
- assert\_count (assert\_numeric), [11](#)
- assert\_decreasing (assert\_constant), [5](#)
- assert\_dim, [6](#), [6](#), [7–15](#)
- assert\_empty, [6](#), [7](#), [7](#), [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, [6](#), [7](#), [8](#), [9–15](#)
- assert\_length, [6–8](#), [8](#), [10–15](#)
- assert\_lengths (assert\_length), [8](#)
- assert\_lower, [6–9](#), [9](#), [10–15](#)
- assert\_missing, [6–10](#), [10](#), [11–15](#)
- assert\_names, [6–10](#), [10](#), [12–15](#)
- assert\_ncol (assert\_dim), [6](#)
- assert\_negative (assert\_numeric), [11](#)
- assert\_nrow (assert\_dim), [6](#)
- assert\_numeric, [6–11](#), [11](#), [13–15](#)
- assert\_odd (assert\_numeric), [11](#)
- assert\_package, [6–12](#), [12](#), [13–15](#)
- assert\_positive (assert\_numeric), [11](#)
- assert\_rownames (assert\_names), [10](#)
- assert\_scalar (assert\_type), [14](#)
- assert\_square, [6–13](#), [13](#), [14](#), [15](#)
- assert\_symmetric (assert\_square), [13](#)
- assert\_type, [6–13](#), [14](#), [15](#)
- assert\_unique, [6–14](#), [15](#)
- 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](#), [36](#), [56](#), [57](#)
- bootstrap(), [24](#)
- bootstrap, numeric-method (bootstrap), [16](#)
- bootstrap-method (bootstrap), [16](#)
- character, [3](#), [5](#), [12](#), [14](#), [17](#), [19](#), [21–23](#), [25](#), [26](#), [32](#), [43](#), [59](#)
- clean\_whitespace, [18](#), [47–55](#)
- clean\_whitespace, data.frame-method (clean\_whitespace), [18](#)
- clean\_whitespace, matrix-method (clean\_whitespace), [18](#)
- clean\_whitespace-method (clean\_whitespace), [18](#)
- compact, [4](#), [5](#), [16](#), [19](#), [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](#), [24–26](#), [34](#), [35](#)
- confidence\_binomial, numeric-method (confidence\_binomial), [21](#)
- confidence\_binomial-method (confidence\_binomial), [21](#)
- confidence\_bootstrap, [22](#), [23](#), [25](#), [26](#), [34](#), [35](#)
- confidence\_bootstrap(), [17](#), [18](#)
- confidence\_bootstrap, numeric-method (confidence\_bootstrap), [23](#)
- confidence\_bootstrap-method (confidence\_bootstrap), [23](#)
- confidence\_mean, [22](#), [24](#), [24](#), [26](#), [34](#), [35](#)
- confidence\_mean, numeric-method (confidence\_mean), [24](#)
- confidence\_mean-method (confidence\_mean), [24](#)
- confidence\_multinomial, [22](#), [24](#), [25](#), [25](#), [34](#), [35](#)
- confidence\_multinomial, numeric-method (confidence\_multinomial), [25](#)
- confidence\_multinomial-method (confidence\_multinomial), [25](#)
- count, [4](#), [5](#), [16](#), [20](#), [27](#), [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](#), [60](#)
- describe, ANY-method (describe), [28](#)
- describe-method (describe), [28](#)
- detect, [4](#), [5](#), [16](#), [20](#), [27](#), [29](#), [31](#), [33](#), [38](#), [59](#)
- detect, ANY-method (detect), [29](#)
- detect-method (detect), [29](#)
- discard, [4](#), [5](#), [16](#), [20](#), [27](#), [30](#), [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, [9](#), [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, [22](#), [24–26](#), [33](#), [35](#)
- interval\_credible, numeric-method (interval\_credible), [33](#)
- interval\_credible-method (interval\_credible), [33](#)
- interval\_hdr, [22](#), [24–26](#), [34](#), [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, [18](#), [36](#), [56](#), [57](#)
- jackknife, numeric-method (jackknife), [36](#)
- jackknife-method (jackknife), [36](#)
- keep, [4](#), [5](#), [16](#), [20](#), [27](#), [30](#), [31](#), [33](#), [37](#), [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, 5, 9, 11, 12, 14, 16, 19, 20, 22, 26, 27, 29, 31, 35, 38, 41–50, 56, 60
- math\_gcd, 39, 40
- math\_gcd,numeric,numeric-method (math\_gcd), 39
- math\_gcd-method (math\_gcd), 39
- math\_lcm, 39, 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, 5, 6, 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, 19, 46, 48–55
- remove\_constant,ANY-method (remove\_constant), 46
- remove\_constant-method (remove\_constant), 46
- remove\_empty, 19, 47, 47, 49–55
- remove\_empty,ANY-method (remove\_empty), 47
- remove\_empty-method (remove\_empty), 47
- remove\_Inf, 19, 47, 48, 48, 50–55
- remove\_Inf,ANY-method (remove\_Inf), 48
- remove\_Inf-method (remove\_Inf), 48
- remove\_NA, 19, 47–49, 49, 51–55
- remove\_NA,ANY-method (remove\_NA), 49
- remove\_NA-method (remove\_NA), 49
- remove\_zero, 19, 47–50, 50, 52–55
- remove\_zero,ANY-method (remove\_zero), 50
- remove\_zero-method (remove\_zero), 50
- replace\_empty, 19, 47–51, 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, 19, 47–52, 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, 19, 47–53, 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, 19, 47–54, 54
- replace\_zero,data.frame-method (replace\_zero), 54
- replace\_zero,matrix-method (replace\_zero), 54
- replace\_zero-method (replace\_zero), 54
- resample\_multinomial, 18, 36, 55, 57
- resample\_multinomial,numeric-method (resample\_multinomial), 55
- resample\_multinomial-method (resample\_multinomial), 55
- resample\_uniform, 18, 36, 56, 56
- resample\_uniform,numeric-method (resample\_uniform), 56
- resample\_uniform-method (resample\_uniform), 56
- scale\_midpoint, 57, 58
- scale\_range, 57, 58
- seek, 4, 5, 16, 20, 27, 30, 31, 33, 38, 58
- 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, 28, 60

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