Package 'arkhe'

January 18, 2023

Title Tools for Cleaning Rectangular Data

Version 1.1.0

Maintainer Nicolas Frerebeau

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

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

License GPL (>= 3)

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

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

Depends R (>= 3.3)

Imports methods, stats, utils

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.2.1

Collate 'AllGenerics.R' 'append.R' 'arkhe-package.R' 'predicates.R' 'assert.R' 'assign.R' 'compact.R' 'conditions.R' 'deprecate.R' 'detect.R' 'discard.R' 'keep.R' 'replace.R' 'reshape.R' 'statistics.R' 'utilities.R' 'zzz.R'

NeedsCompilation no

Author Nicolas Frerebeau [aut, cre] (<https://orcid.org/0000-0001-5759-4944>, Université Bordeaux Montaigne), Brice Lebrun [ctb] (<https://orcid.org/0000-0001-7503-8685>, Université Bordeaux Montaigne)

Repository CRAN

Date/Publication 2023-01-18 09:00:15 UTC

R topics documented:

---

append *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 column to use for row names.

Value

A data.frame.

Author(s)

N. Frerebeau

See Also

Other transformation tools: assign(), reshape()

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))
```

---

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 transformation tools: append(), reshape()

Examples

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

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

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

bootstrap *Bootstrap Estimation*

Description

Samples randomly from the elements of `object` with replacement.

Usage

```
bootstrap(object, ...)

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

Arguments

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

Value

If `f` is not `NULL`, `bootstrap()` returns the result of `f` applied to the `n` values of `do`.

If `f` is `NULL`, `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` he bootstrap estimate of standard error of `do`.

Author(s)

N. Frerebeau

See Also

Other resampling methods: `jackknife()`

Examples

```
x <- rnorm(20)

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

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

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

---

check-attribute *Check Object Attributes*

Description

Check Object Attributes

Usage

```
assert_empty(x)

assert_filled(x)

assert_length(x, expected, empty = FALSE)

assert_lengths(x, expected)

assert_dimensions(x, expected)

assert_names(x, expected)

assert_dimnames(x, expected)

assert_rownames(x, expected)

assert_colnames(x, expected)
```

Arguments

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

Value

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

Author(s)

N. Frerebeau

See Also

Other validation methods: `check-data`, `check-matrix`, `check-numeric-comparison`, `check-numeric-trend`, `check-numeric`, `check-type`, `needs()`, `validate()`

`check-data` *Check Data*

Description

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

Usage

```
assert_missing(x)

assert_infinite(x)

assert_unique(x)
```

Arguments

`x`	An object to be checked.

Value

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

Author(s)

N. Frerebeau

See Also

Other validation methods: `check-attribute`, `check-matrix`, `check-numeric-comparison`, `check-numeric-trend`, `check-numeric`, `check-type`, `needs()`, `validate()`

check-matrix *Check Matrix*

Description

Check Matrix

Usage

```
assert_symmetric(x)

assert_square(x)
```

Arguments

x	A matrix to be checked.

Value

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

Author(s)

N. Frerebeau

See Also

Other validation methods: check-attribute, check-data, check-numeric-comparison, check-numeric-trend, check-numeric, check-type, needs(), validate()

check-numeric *Check Numeric Values*

Description

Check Numeric Values

Usage

```
assert_count(x, ...)

assert_whole(x, ...)

assert_positive(x, ...)

assert_negative(x, ...)
```

```
assert_odd(x, ...)

assert_even(x, ...)
```

Arguments

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

Value

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

Author(s)

N. Frerebeau

See Also

Other validation methods: check-attribute, check-data, check-matrix, check-numeric-comparison, check-numeric-trend, check-type, needs(), validate()

---

check-numeric-comparison

Check Numeric Relations

Description

Check Numeric Relations

Usage

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

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

Arguments

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

Value

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

Author(s)

N. Frerebeau

See Also

Other validation methods: check-attribute, check-data, check-matrix, check-numeric-trend, check-numeric, check-type, needs(), validate()

---

check-numeric-trend *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 validation methods: check-attribute, check-data, check-matrix, check-numeric-comparison, check-numeric, check-type, needs(), validate()

check-type *Check Data Types*

Description

Check Data Types

Usage

```
assert_type(x, expected)

assert_scalar(x, expected)
```

Arguments

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

Value

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

Author(s)

N. Frerebeau

See Also

Other validation methods: check-attribute, check-data, check-matrix, check-numeric-comparison, check-numeric-trend, check-numeric, needs(), validate()

compact *Remove empty rows/columns*

Description

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

Usage

```
compact(x, ...)

compact_cols(x, ...)

compact_rows(x, ...)

## S4 method for signature 'ANY'
compact(x, margin = 1)

## S4 method for signature 'ANY'
compact_cols(x)

## S4 method for signature 'ANY'
compact_rows(x)
```

Arguments

x	An object (should be a matrix or a data.frame).
...	Currently not used.
margin	A vector giving the subscripts which the function will be applied over (1 indicates rows, 2 indicates columns).

Details

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

Author(s)

N. Frerebeau

See Also

Other data cleaning tools: count(), detect(), discard(), infinite, keep(), missing, zero

Examples

```
## Create a count 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)

## 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)

## 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)

## 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)

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

`confidence_binomial` *Confidence Interval for Binomial Proportions*

Description

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

Usage

```
confidence_binomial(object, ...)

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

Arguments

`object`	A `numeric` vector giving the number of success.
`...`	Currently not used.
`n`	A length-one `numeric` vector giving the number of trials.
`level`	A length-one `numeric` vector giving the confidence level. Must be a single number between 0 and 1.

method	A character string specifying the method to be used. Any unambiguous substring can be used.
corrected	A logical scalar: should continuity correction be used? Only used if method is "wald".

Value

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

Author(s)

N. Frerebeau

See Also

Other summary statistics: confidence_mean(), confidence_multinomial()

Examples

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

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

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

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

---

confidence_mean *Confidence Interval for a Mean*

Description

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

Usage

```
confidence_mean(object, ...)

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

Arguments

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

Value

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

Author(s)

N. Frerebeau

See Also

Other summary statistics: confidence_binomial(), confidence_multinomial()

Examples

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

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

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

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

---

confidence_multinomial

Confidence Interval for Multinomial Proportions

Description

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

Usage

```
confidence_multinomial(object, ...)

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

Arguments

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_mean()

Examples

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

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

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

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

count *Count values using a predicate*

Description

Counts values by rows/columns using a predicate function.

Usage

```
count(x, f, ...)

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

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

Arguments

x	An object (should be a matrix or a data.frame).
f	A predicate function.
...	Currently not used.
margin	A 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?

Value

A numeric vector.

Author(s)

N. Frerebeau

See Also

Other data cleaning tools: compact(), detect(), discard(), infinite, keep(), missing, zero

Examples

```
## Create a count 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)

## 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)

## 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)

## 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)

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

`detect` *Find rows/columns using a predicate*

Description

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

Usage

```
detect(x, f, ...)

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

Arguments

`x`	An object (should be a `matrix` or a `data.frame`).
`f`	A predicate `function`.
`...`	Currently not used.
`margin`	A 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.

Value

A logical vector.

Author(s)

N. Frerebeau

See Also

Other data cleaning tools: compact(), count(), discard(), infinite, keep(), missing, zero

Examples

```
## Create a count 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)

## 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)

## 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)

## 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)

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

---

discard *Remove rows/columns using a predicate*

Description

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

Usage

```
discard(x, f, ...)

discard_cols(x, f, ...)

discard_rows(x, f, ...)

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

## S4 method for signature 'ANY,`function`'
discard_rows(x, f, negate = FALSE, all = FALSE)

## S4 method for signature 'ANY,`function`'
discard_cols(x, f, negate = FALSE, all = FALSE)
```

Arguments

x	An object (should be a matrix or a data.frame).
f	A predicate function.
...	Currently not used.
margin	A 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.

Author(s)

N. Frerebeau

See Also

Other data cleaning tools: compact(), count(), detect(), infinite, keep(), missing, zero

Examples

```
## Create a count 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)

## 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)

## 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)

## 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)

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

`infinite` *Tools for working with infinite values*

Description

- `remove_Inf()` remove rows/columns that contain infinite values.
- `replace_Inf` replaces infinite values values.

Usage

```
remove_Inf(x, ...)

replace_Inf(x, ...)

## S4 method for signature 'ANY'
remove_Inf(x, margin = 1, all = FALSE)

## S4 method for signature 'matrix'
replace_Inf(x, value = 0)
```

Arguments

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

Author(s)

N. Frerebeau

See Also

Other data cleaning tools: compact(), count(), detect(), discard(), keep(), missing, zero

Examples

```
## Create a count 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)

## 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)

## 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)

## 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)

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

jackknife *Jackknife Estimation*

Description

Jackknife Estimation

Usage

```
jackknife(object, ...)

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

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.

Value

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 he jackknife estimate of standard error of do.

Author(s)

N. Frerebeau

See Also

Other resampling methods: bootstrap()

Examples

```
x <- rnorm(20)

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

## Estimate the 25th and 95th percentiles
quant <- function(x) { quantile(x, probs = c(0.25, 0.75)) }
```

```
bootstrap(x, n = 100, do = mean, f = quant)

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

---

keep *Keep rows/columns using a predicate*

Description

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

Usage

```
keep(x, f, ...)

keep_cols(x, f, ...)

keep_rows(x, f, ...)

## S4 method for signature 'ANY,`function`'
keep(x, f, margin = 1, negate = FALSE, all = FALSE)

## S4 method for signature 'ANY,`function`'
keep_rows(x, f, negate = FALSE, all = FALSE)

## S4 method for signature 'ANY,`function`'
keep_cols(x, f, negate = FALSE, all = FALSE)
```

Arguments

x	An object (should be a matrix or a data.frame).
f	A predicate function.
...	Currently not used.
margin	A 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.

Author(s)

N. Frerebeau

See Also

Other data cleaning tools: compact(), count(), detect(), discard(), infinite, missing, zero

Examples

```
## Create a count 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)

## 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)

## 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)

## 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)

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

---

missing *Tools for working with missing values*

Description

- remove_NA() remove rows/columns that contain missing values.
- replace_NA replaces missing values values.

Usage

```
remove_NA(x, ...)

replace_NA(x, ...)
```

```
## S4 method for signature 'ANY'
remove_NA(x, margin = 1, all = FALSE)

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

Arguments

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

Author(s)

N. Frerebeau

See Also

Other data cleaning tools: compact(), count(), detect(), discard(), infinite, keep(), zero

Examples

```
## Create a count 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)

## 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)

## 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)
```

```
## 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)

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

---

needs *Check the Availability of a Package*

Description

Check the Availability of a Package

Usage

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

Arguments

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

Details

needs() 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 validation methods: check-attribute, check-data, check-matrix, check-numeric-comparison, check-numeric-trend, check-numeric, check-type, validate()

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: `predicate-numeric`, `predicate-scalar`, `predicate-trend`, `predicate-type`, `predicate-utils`

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, na.rm = FALSE)

is_odd(x, na.rm = FALSE)

is_even(x, na.rm = FALSE)

is_positive(x, strict = FALSE, na.rm = FALSE)

is_negative(x, strict = FALSE, na.rm = FALSE)

is_whole(x, na.rm = FALSE, tolerance = .Machine$double.eps^0.5)
```

Arguments

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

Value

A logical vector.

See Also

Other predicates: predicate-matrix, predicate-scalar, predicate-trend, predicate-type, predicate-utils

---

predicate-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-matrix, predicate-numeric, predicate-trend, predicate-type, predicate-utils

---

predicate-trend *Numeric Trend Predicates*

Description

Check numeric objects:

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

Usage

```
is_constant(x, tolerance = .Machine$double.eps^0.5, 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: predicate-matrix, predicate-numeric, predicate-scalar, predicate-type, predicate-utils

---

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: predicate-matrix, predicate-numeric, predicate-scalar, predicate-trend, predicate-utils

---

predicate-utils *Utility Predicates*

Description

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

Usage

```
has_length(x, n = NULL)

has_names(x, names = NULL)

has_duplicates(x)

has_missing(x)

has_infinite(x)

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.
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: predicate-matrix, predicate-numeric, predicate-scalar, predicate-trend, predicate-type

---

reshape *Reshape*

Description

Transforms a matrix to a long data.frame.

Usage

```
wide_to_long(from, ...)

to_long(from, ...)

## S4 method for signature 'matrix'
wide_to_long(from, factor = FALSE, reverse = FALSE)

## S4 method for signature 'matrix'
to_long(from, factor = FALSE, reverse = FALSE)
```

Arguments

from	An object to be coerced.
...	Currently not used.
factor	A logical scalar: should character string be coerced to factor? Default to FALSE, if TRUE the original ordering is preserved.
reverse	A logical scalar: should the order of factor levels be reversed? Only used if factor is TRUE. Useful for plotting.

Value

A coerced object.

Author(s)

N. Frerebeau

See Also

Other transformation tools: append(), assign()

Examples

```
## Create a matrix
A <- matrix(data = sample(0:10, 100, TRUE), nrow = 20, ncol = 5)

## Transform to long data.frame
head(wide_to_long(A))
```

`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

See Also

Other validation methods: `check-attribute`, `check-data`, `check-matrix`, `check-numeric-comparison`, `check-numeric-trend`, `check-numeric`, `check-type`, `needs()`

zero *Tools for working with zeros*

Description

- `remove_zero()` remove rows/columns that contain zeros.
- `replace_zero` replaces zeros.

Usage

```
remove_zero(x, ...)

replace_zero(x, ...)

## S4 method for signature 'ANY'
remove_zero(x, margin = 1, all = FALSE)

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

Arguments

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

Author(s)

N. Frerebeau

See Also

Other data cleaning tools: `compact()`, `count()`, `detect()`, `discard()`, `infinite`, `keep()`, `missing`

Examples

```
## Create a count 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)

## 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)

## 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)

## 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)

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

Index