Package 'arkhe'

June 15, 2022

Title Representation of Archaeological Data

Version 0.5.0

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

Description A collection of classes that represent archaeological data. This package provides a set of S4 classes that represent different special types of matrix (absolute/relative frequency, presence/absence data, co-occurrence matrix, etc.) upon which package developers can build subclasses. It also provides a set of generic methods (mutators and coercion mechanisms) and functions (e.g. summary statistics, predicates). In addition, a few classes of general interest (e.g. that represent stratigraphic relationships) are implemented.

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 covr, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.2.0

Collate 'AllClasses.R' 'AllGenerics.R' 'arkhe-package.R' 'assign.R' 'predicates.R' 'check.R' 'coerce.R' 'compact.R' 'conditions.R' 'detect.R' 'initialize.R' 'mutators.R' 'replace.R' 'show.R' 'statistics.R' 'subset.R' 'summary.R' 'utilities.R' 'validate.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 2022-06-15 09:30:12 UTC

R topics documented:

assign *Assign a specific row/column to the column/row names*

Description

Assign a specific row/column to the column/row names

Usage

```
assign_colnames(x, ...)

assign_rownames(x, ...)

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

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

Arguments

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

Value

A data.frame.

Author(s)

N. Frerebeau

See Also

Other data cleaning tools: count(), detect(), remove(), replace()

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)
```

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-graph, check-matrix, check-numeric, check-type, 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, 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 validation methods: `check-attribute`, `check-graph`, `check-matrix`, `check-numeric`, `check-type`, `validate()`

check-matrix *Check Matrix*

Description

Check Matrix

Usage

```
assert_matrix(x, expected)
```

Arguments

x	A matrix to be checked.
expected	A character string specifying the expected value. It must be one of "square" or "symmetric".

Value

Throw an error, if any.

Author(s)

N. Frerebeau

See Also

Other validation methods: check-attribute, check-data, check-graph, check-numeric, check-type, validate()

---

check-numeric *Check Numeric Values*

Description

Check Numeric Values

Usage

```
assert_count(x)

assert_numeric(x, expected, ...)

assert_trend(x, expected, ...)

assert_relation(x, y, expected, ...)
```

Arguments

x, y	A numeric object to be checked.
expected	A character string specifying the expected value (see details).
...	Extra parameters to be passed to internal methods.

Details

Possible values for expected:

assert_numeric() "positive", "whole", "odd" or "even"

assert_trend() "constant", "decreasing" or "increasing"

assert_relation() "lower" or "greater"

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-graph, check-matrix, check-type, 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-graph, check-matrix, check-numeric, validate()

coerce *Coerce*

Description

Coerces an object to a *Matrix object.

Usage

```
as_long(from, ...)

as_count(from)

as_composition(from)

as_incidence(from)

as_occurrence(from)

as_features(from)

as_stratigraphy(from)

## S4 method for signature 'ANY'
as_count(from)

## S4 method for signature 'ANY'
as_composition(from)

## S4 method for signature 'ANY'
as_incidence(from)

## S4 method for signature 'ANY'
as_occurrence(from)

## S4 method for signature 'ANY'
as_stratigraphy(from)

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

## S4 method for signature 'AbundanceMatrix'
as_long(from, factor = FALSE, reverse = FALSE)

## S4 method for signature 'AbundanceMatrix'
as_features(from)
```

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.

Details

The following methods coerce an object to a *Matrix object:

Method	Target	Details
as_count()	CountMatrix	absolute frequency data
as_composition()	CompositionMatrix	relative frequency data
as_incidence()	IncidenceMatrix	presence/absence data
as_occurrence()	OccurrenceMatrix	co-occurrence
as_stratigraphy()	StratigraphicMatrix	stratigraphic relationships

Note that as_count() *rounds numeric values to zero decimal places and then coerces to integer as by* as.integer().

as_stratigraphy() converts a set of stratigraphic relationships (edges) to a stratigraphic (adjacency) matrix. from can be a matrix, list, or data.frame: the first column/component is assumed to contain the bottom units and the second the top units (adjacency).

Method	Target	Details
as_long()	data.frame	long S3 data frame
as_features()	data.frame	wide S3 data frame

as_features() converts a *Matrix object to a collection of features: a data.frame with all informations as extra columns (result may differ according to the class of from).

Value

A coerced object.

Abundance Matrix

The CountMatrix, CompositionMatrix and IncidenceMatrix classes have special slots:

- samples for replicated measurements/observation,
- groups to group data by site/area,
- dates to specify the date point estimate of an assemblage,
- tqp and taq to specify the chronology of an assemblage.

When coercing a `data.frame` to a `*Matrix` object, an attempt is made to automatically assign values to these slots by mapping column names (case insensitive, plural insensitive). This behavior can be disabled by setting `options(arkhe.autodetect = FALSE)` or overrided by explicitly specifying the columns to be used in `as_*()`.

Chronology

The way chronological information is handled is somewhat opinionated. Sub-annual precision is overkill/meaningless in most situations: dates are assumed to be expressed in years CE and are stored as integers (values are coerced with `as.integer()` and hence truncated towards zero).

Author(s)

N. Frerebeau

See Also

Other classes: `CompositionMatrix-class`, `CountMatrix-class`, `DataMatrix`, `IncidenceMatrix-class`, `OccurrenceMatrix-class`, `StratigraphicMatrix-class`

Examples

```
## Create a count matrix
A0 <- matrix(data = sample(0:10, 100, TRUE), nrow = 20, ncol = 5)

## Coerce to absolute frequencies
A1 <- as_count(A0)

## Coerce to relative frequencies
B0 <- as_composition(A1)

## Row sums are internally stored before coercing to relative frequencies
## (use get_totals() to retrieve these values)
## This allows to restore the source data
A2 <- as_count(B0)
all(A1 == A2)

## Coerce to presence/absence
C0 <- as_incidence(A1)

## Coerce to a co-occurrence matrix
D0 <- as_occurrence(A1)

## Coerce to an S3 matrix or data.frame
X <- as.matrix(A1)
all(A0 == X)

Y <- data.frame(A1)
head(Y)
```

CompositionMatrix-class

Relative Frequency Matrix

Description

An S4 class to represent a relative frequency matrix (i.e. the fraction of times a given datum occurs in a dataset).

Usage

```
CompositionMatrix(data = 0, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)
```

Arguments

data	an optional data vector (including a list or expression vector). Non-atomic classed R objects are coerced by as.vector and all attributes discarded.
nrow	the desired number of rows.
ncol	the desired number of columns.
byrow	logical. If FALSE (the default) the matrix is filled by columns, otherwise the matrix is filled by rows.
dimnames	A dimnames attribute for the matrix: NULL or a list of length 2 giving the row and column names respectively. An empty list is treated as NULL, and a list of length one as row names. The list can be named, and the list names will be used as names for the dimensions.

Author(s)

N. Frerebeau

See Also

as_composition()

Other classes: CountMatrix-class, DataMatrix, IncidenceMatrix-class, OccurrenceMatrix-class, StratigraphicMatrix-class, coerce()

Examples

```
## Create an incidence (presence/absence) matrix
## Data will be coerced with as.logical()
A <- IncidenceMatrix(data = sample(0:1, 100, TRUE, c(1, 1/3)), nrow = 20)
## Create a count data matrix
B <- CountMatrix(data = sample(0:10, 100, TRUE), nrow = 20)

## Access
dim(B) # Get the matrix dimensions
row(B) # Get the row indexes
```

```
col(B, as.factor = TRUE) # Get the column indexes
nrow(B) # Get the number of rows
ncol(B) # Get the number of columns
dimnames(B) # Get the dimension names
rownames(B) <- LETTERS[1:20] # Set the row names
rownames(B) # Get the rownames
colnames(B) <- letters[21:25] # Set the column names
colnames(B) # Get the column names

## Subset
B[[1, 1]] # Get the first value
B[1] # Get the first value
B[, ] # Get all values
B[1, , drop = FALSE] # Get the first row
B[, 1:3] # Get the first three column
```

`confidence` *Confidence Interval for a Mean*

Description

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

Usage

```
confidence(object, ...)

## S4 method for signature 'numeric'
confidence(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 lower and upper confidence limits.

Author(s)

N. Frerebeau

See Also

Other resampling methods: jackknife()

Examples

```
## Jackknife
x <- rnorm(20)
jackknife(x, do = mean) # Sample mean
```

---

count *Count values according to a given predicate*

Description

Counts values by rows/columns according to a given predicate.

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: assign(), detect(), remove(), replace()

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)

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

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

CountMatrix-class *Absolute Frequency Matrix*

Description

An S4 class to represent an absolute frequency matrix (i.e. the number of times a given datum occurs in a dataset).

Usage

```
CountMatrix(data = 0, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)
```

Arguments

data	an optional data vector (including a list or expression vector). Non-atomic classed R objects are coerced by as.vector and all attributes discarded.
nrow	the desired number of rows.
ncol	the desired number of columns.
byrow	logical. If FALSE (the default) the matrix is filled by columns, otherwise the matrix is filled by rows.

dimnames — A dimnames attribute for the matrix: NULL or a list of length 2 giving the row and column names respectively. An empty list is treated as NULL, and a list of length one as row names. The list can be named, and the list names will be used as names for the dimensions.

Author(s)

N. Frerebeau

See Also

as_count()

Other classes: CompositionMatrix-class, DataMatrix, IncidenceMatrix-class, OccurrenceMatrix-class, StratigraphicMatrix-class, coerce()

Examples

```
## Create an incidence (presence/absence) matrix
## Data will be coerced with as.logical()
A <- IncidenceMatrix(data = sample(0:1, 100, TRUE, c(1, 1/3)), nrow = 20)
## Create a count data matrix
B <- CountMatrix(data = sample(0:10, 100, TRUE), nrow = 20)

## Access
dim(B) # Get the matrix dimensions
row(B) # Get the row indexes
col(B, as.factor = TRUE) # Get the column indexes
nrow(B) # Get the number of rows
ncol(B) # Get the number of columns
dimnames(B) # Get the dimension names
rownames(B) <- LETTERS[1:20] # Set the row names
rownames(B) # Get the rownames
colnames(B) <- letters[21:25] # Set the column names
colnames(B) # Get the column names

## Subset
B[[1, 1]] # Get the first value
B[1] # Get the first value
B[, ] # Get all values
B[1, , drop = FALSE] # Get the first row
B[, 1:3] # Get the first three column
```

---

detect *Find values according to a given predicate*

Description

Finds rows/columns in an array-like object according to a given predicate.

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: assign(), count(), remove(), replace()

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)
```

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

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

`IncidenceMatrix-class` *Incidence Matrix*

Description

An S4 class to represent an incidence (presence/absence) matrix.

Usage

```
IncidenceMatrix(
  data = FALSE,
  nrow = 1,
  ncol = 1,
  byrow = FALSE,
  dimnames = NULL
)
```

Arguments

data	an optional data vector (including a list or expression vector). Non-atomic classed R objects are coerced by as.vector and all attributes discarded.
nrow	the desired number of rows.
ncol	the desired number of columns.
byrow	logical. If FALSE (the default) the matrix is filled by columns, otherwise the matrix is filled by rows.
dimnames	A dimnames attribute for the matrix: NULL or a list of length 2 giving the row and column names respectively. An empty list is treated as NULL, and a list of length one as row names. The list can be named, and the list names will be used as names for the dimensions.

Author(s)

N. Frerebeau

See Also

as_incidence()

Other classes: CompositionMatrix-class, CountMatrix-class, DataMatrix, OccurrenceMatrix-class, StratigraphicMatrix-class, coerce()

Examples

```
## Create an incidence (presence/absence) matrix
## Data will be coerced with as.logical()
A <- IncidenceMatrix(data = sample(0:1, 100, TRUE, c(1, 1/3)), nrow = 20)
## Create a count data matrix
B <- CountMatrix(data = sample(0:10, 100, TRUE), nrow = 20)

## Access
dim(B) # Get the matrix dimensions
row(B) # Get the row indexes
col(B, as.factor = TRUE) # Get the column indexes
nrow(B) # Get the number of rows
ncol(B) # Get the number of columns
dimnames(B) # Get the dimension names
rownames(B) <- LETTERS[1:20] # Set the row names
rownames(B) # Get the rownames
colnames(B) <- letters[21:25] # Set the column names
colnames(B) # Get the column names

## Subset
B[[1, 1]] # Get the first value
B[1] # Get the first value
B[, ] # Get all values
B[1, , drop = FALSE] # Get the first row
B[, 1:3] # Get the first three column
```

---

jackknife *Jackknife Estimation*

Description

Jackknife Estimation

Usage

```
jackknife(object, ...)

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

Arguments

object	A numeric vector.
...	Extra arguments 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:

`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: `confidence()`

Examples

```
## Jackknife
x <- rnorm(20)
jackknife(x, do = mean) # Sample mean
```

---

mutators *Get or Set Parts of an Object*

Description

Getters and setters to retrieve or set parts of an object.

Usage

```
has_groups(x)

get_groups(x)

set_groups(x) <- value

get_samples(x)

set_samples(x) <- value

has_dates(x)

get_dates(x)

set_dates(x) <- value

has_terminus(x)
```

```
get_terminus(x)

set_terminus(x) <- value

get_tpq(x)

set_tpq(x) <- value

get_taq(x)

set_taq(x) <- value

get_totals(x)

set_totals(x) <- value

## S4 method for signature 'AbundanceMatrix'
has_groups(x)

## S4 method for signature 'AbundanceMatrix'
get_groups(x)

## S4 method for signature 'AbundanceMatrix'
get_samples(x)

## S4 method for signature 'AbundanceMatrix'
has_dates(x)

## S4 method for signature 'AbundanceMatrix'
get_dates(x)

## S4 method for signature 'AbundanceMatrix'
has_terminus(x)

## S4 method for signature 'AbundanceMatrix'
get_terminus(x)

## S4 method for signature 'AbundanceMatrix'
get_tpq(x)

## S4 method for signature 'AbundanceMatrix'
get_taq(x)

## S4 method for signature 'AbundanceMatrix'
get_totals(x)

## S4 method for signature 'OccurrenceMatrix'
```

```
get_totals(x)

## S4 replacement method for signature 'AbundanceMatrix'
set_groups(x) <- value

## S4 replacement method for signature 'AbundanceMatrix'
set_samples(x) <- value

## S4 replacement method for signature 'AbundanceMatrix,`NULL`'
set_dates(x) <- value

## S4 replacement method for signature 'AbundanceMatrix,numeric'
set_dates(x) <- value

## S4 replacement method for signature 'AbundanceMatrix,`NULL`'
set_terminus(x) <- value

## S4 replacement method for signature 'AbundanceMatrix,list'
set_terminus(x) <- value

## S4 replacement method for signature 'AbundanceMatrix,`NULL`'
set_tpq(x) <- value

## S4 replacement method for signature 'AbundanceMatrix,numeric'
set_tpq(x) <- value

## S4 replacement method for signature 'AbundanceMatrix,`NULL`'
set_taq(x) <- value

## S4 replacement method for signature 'AbundanceMatrix,numeric'
set_taq(x) <- value

## S4 replacement method for signature 'AbundanceMatrix'
set_totals(x) <- value
```

Arguments

x	An object from which to get or set element(s) (typically an AbundanceMatrix object).
value	A possible value for the element(s) of x.

Details

`get_samples(x)` **and** `get_samples(x) <- value` Get or set the sample names of `x`.

`get_groups(x)` **and** `set_groups(x) <- value` Get or set the groups of `x`.

`get_dates(x)` **and** `set_dates(x) <- value` Get or set the dates of `x`.

`get_terminus(x)` **and** `set_terminus(x) <- value` Get or set the chronology of `x`. `value` must be a `list` with components `tpq` (TPQ - *terminus post quem*) and `taq` (TAQ - *terminus ante*

quem).

`get_tpq(x)` **and** `set_tpq(x) <- value`**,** `get_taq(x)` **and** `set_taq(x) <- value` Get or set the TPQ/TAQ of `x`.

Value

- `set_*()` returns an object of the same sort as `x` with the new values assigned.
- `get_*()` returns the part of `x`.
- `has_*()` returns a `logical` scalar.

Chronology

The way chronological information is handled is somewhat opinionated. Sub-annual precision is overkill/meaningless in most situations: dates are assumed to be expressed in years CE and are stored as integers (values are coerced with `as.integer()` and hence truncated towards zero).

Author(s)

N. Frerebeau

See Also

Other mutators: `subset()`, `summary()`

Examples

```
## Create a data.frame
X <- matrix(data = sample(0:10, 50, TRUE), nrow = 10, ncol = 5)
Y <- as.data.frame(X)

## Coerce to a count matrix
Z <- as_count(Y)

## Set/get groups
set_samples(Z) <- rep(c("a", "b", "c", "d", "e"), each = 2)
get_samples(Z)

## Set/get groups
set_groups(Z) <- rep(c("A", "B"), each = 5)
get_groups(Z)

## Get/get TPQ/TAQ
chrono <- list(
  tpq = sample(1301:1400, 10, replace = TRUE),
  taq = sample(1451:1500, 10, replace = TRUE)
)
set_terminus(Z) <- chrono
get_terminus(Z)

## Collection of features
as_features(Z)
```

```
## Summarize data
summary(Z)
```

OccurrenceMatrix-class

Co-Occurrence Matrix

Description

An S4 class to represent a co-occurrence matrix.

Details

A co-occurrence matrix is a symmetric matrix with zeros on its main diagonal, which works out how many times each pairs of taxa/types occur together in at least one sample.

Slots

`total` An `integer` giving the total number of observations.

Author(s)

N. Frerebeau

See Also

`as_occurrence()`

Other classes: `CompositionMatrix-class`, `CountMatrix-class`, `DataMatrix`, `IncidenceMatrix-class`, `StratigraphicMatrix-class`, `coerce()`

Examples

```
## Create an incidence (presence/absence) matrix
## Data will be coerced with as.logical()
A <- IncidenceMatrix(data = sample(0:1, 100, TRUE, c(1, 1/3)), nrow = 20)
## Create a count data matrix
B <- CountMatrix(data = sample(0:10, 100, TRUE), nrow = 20)

## Access
dim(B) # Get the matrix dimensions
row(B) # Get the row indexes
col(B, as.factor = TRUE) # Get the column indexes
nrow(B) # Get the number of rows
ncol(B) # Get the number of columns
dimnames(B) # Get the dimension names
rownames(B) <- LETTERS[1:20] # Set the row names
rownames(B) # Get the rownames
colnames(B) <- letters[21:25] # Set the column names
```

```
colnames(B) # Get the column names

## Subset
B[[1, 1]] # Get the first value
B[1] # Get the first value
B[, ] # Get all values
B[1, , drop = FALSE] # Get the first row
B[, 1:3] # Get the first three column
```

`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-graph`, `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-graph, 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-graph, 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-graph, 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-graph, 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-graph, predicate-matrix, predicate-numeric, predicate-scalar, predicate-trend, predicate-type

---

remove *Remove values according to a given predicate*

Description

Removes rows/columns in an array-like object according to a given predicate.

Usage

```
compact(x, f, ...)

remove_NA(x, ...)

remove_Inf(x, ...)

remove_zero(x, ...)

remove_empty(x, ...)

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

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

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

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

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

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.

Details

- remove_NA() remove rows/columns that contain missing values.
- remove_Inf() remove rows/columns that contain infinite values.
- remove_zero() remove rows/columns that contain zero.
- remove_empty() is a special case that remove empty rows/columns. 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: assign(), count(), detect(), replace()

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)

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

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

`replace` *Data Replacement*

Description

Replaces `missing` or `infinite` values or zeros.

Usage

```
replace_NA(x, ...)

replace_Inf(x, ...)

replace_zero(x, ...)

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

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

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

Arguments

`x`	A `matrix`, a `data.frame` or a `*Matrix` object.
`...`	Currently not used.
`value`	A possible value to replace missing or infinite values of `x`.

Author(s)

N. Frerebeau

See Also

Other data cleaning tools: `assign()`, `count()`, `detect()`, `remove()`

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)

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

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

StratigraphicMatrix-class

Stratigraphic Matrix

Description

An S4 class to represent a stratigraphic matrix.

Details

A stratigraphic matrix represents directed relationships between stratigraphic units. A stratigraphic matrix is an adjacency matrix (a non symmetric square matrix with zeros on its main diagonal), suitable to build a directed acyclic graph (DAG).

Author(s)

N. Frerebeau

See Also

`as_stratigraphy()`

Other classes: `CompositionMatrix-class`, `CountMatrix-class`, `DataMatrix`, `IncidenceMatrix-class`, `OccurrenceMatrix-class`, `coerce()`

Examples

```
# Principles of Archaeological Stratigraphy, fig. 12
harris <- matrix(
  data = c(2, 1,
           3, 1,
           4, 1,
           5, 2,
           5, 3,
           5, 4,
           6, 5,
           7, 1,
           7, 6,
           8, 1,
           8, 6,
           9, 7,
           9, 8),
  ncol = 2,
  byrow = TRUE,
  dimnames = list(NULL, c("lower", "upper"))
)

strati <- as_stratigraphy(harris)
```

---

subset *Extract or Replace Parts of an Object*

Description

Operators acting on objects to extract or replace parts.

Usage

```
## S4 method for signature 'AbundanceMatrix'
x[i, j, ..., drop = TRUE]

## S4 replacement method for signature 'AbundanceMatrix'
x[i, j, ...] <- value

## S4 replacement method for signature 'AbundanceMatrix'
x[[i, j, ...]] <- value
```

Arguments

x	An object from which to extract element(s) or in which to replace element(s) (typically a *Matrix object).
i, j	Indices specifying elements to extract or replace. Indices are numeric, integer or character vectors or empty (missing) or NULL. Numeric values are coerced to integer as by as.integer() (and hence truncated towards zero). Character

	vectors will be matched to the name of the elements. An empty index (a comma separated blank) indicates that all entries in that dimension are selected.
...	Currently not used.
drop	A logical scalar: should the result be coerced to the lowest possible dimension? This only works for extracting elements, not for the replacement.
value	A possible value for the element(s) of x.

Value

A subsetted object of the same sort as x.

Author(s)

N. Frerebeau

See Also

Other mutators: mutators, summary()

Examples

```
## Create an incidence (presence/absence) matrix
## Data will be coerced with as.logical()
A <- IncidenceMatrix(data = sample(0:1, 100, TRUE, c(1, 1/3)), nrow = 20)
## Create a count data matrix
B <- CountMatrix(data = sample(0:10, 100, TRUE), nrow = 20)

## Access
dim(B) # Get the matrix dimensions
row(B) # Get the row indexes
col(B, as.factor = TRUE) # Get the column indexes
nrow(B) # Get the number of rows
ncol(B) # Get the number of columns
dimnames(B) # Get the dimension names
rownames(B) <- LETTERS[1:20] # Set the row names
rownames(B) # Get the rownames
colnames(B) <- letters[21:25] # Set the column names
colnames(B) # Get the column names

## Subset
B[[1, 1]] # Get the first value
B[1] # Get the first value
B[, ] # Get all values
B[1, , drop = FALSE] # Get the first row
B[, 1:3] # Get the first three column
```

summary *Object Summaries*

Description

Produces result summaries.

Usage

```
## S4 method for signature 'AbundanceMatrix'
summary(object, ...)
```

Arguments

object	An AbundanceMatrix object.
...	Currently not used.

Value

An AbundanceSummary object.

Author(s)

N. Frerebeau

See Also

Other mutators: mutators, subset()

Examples

```
## Create a data.frame
X <- matrix(data = sample(0:10, 50, TRUE), nrow = 10, ncol = 5)
Y <- as.data.frame(X)

## Coerce to a count matrix
Z <- as_count(Y)

## Set/get groups
set_samples(Z) <- rep(c("a", "b", "c", "d", "e"), each = 2)
get_samples(Z)

## Set/get groups
set_groups(Z) <- rep(c("A", "B"), each = 5)
get_groups(Z)

## Get/get TPQ/TAQ
chrono <- list(
  tpq = sample(1301:1400, 10, replace = TRUE),
```

```
  taq = sample(1451:1500, 10, replace = TRUE)
)
set_terminus(Z) <- chrono
get_terminus(Z)

## Collection of features
as_features(Z)

## Summarize data
summary(Z)
```

---

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-graph, check-matrix, check-numeric, check-type

Index