Package 'arkhe'

September 18, 2021

Title Representation of Archaeological Data

Version 0.4.0

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 7.1.1

Collate 'AllClasses.R' 'AllGenerics.R' 'arkhe-package.R' 'predicates.R' 'check.R' 'clean.R' 'coerce.R' 'conditions.R' 'initialize.R' 'mutators.R' 'show.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>),
Brice Lebrun [ctb] (<https://orcid.org/0000-0001-7503-8685>)

Repository CRAN

Date/Publication 2021-09-18 14:20:02 UTC

R topics documented:

check-attribute . . . 2
check-data . . . 3
check-matrix . . . 4
check-numeric . . . 5
check-type . . . 6
coerce . . . 6
CompositionMatrix-class . . . 9
CountMatrix-class . . . 11
IncidenceMatrix-class . . . 12
mutators . . . 13
OccurrenceMatrix-class . . . 17
predicate-matrix . . . 18
predicate-numeric . . . 19
predicate-scalar . . . 20
predicate-trend . . . 20
predicate-type . . . 21
predicate-utils . . . 22
remove . . . 23
replace . . . 24
StratigraphicMatrix-class . . . 26
subset . . . 27
summary . . . 28
validate . . . 29

Index **31**

`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_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
```

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
```

`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
```

---

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 *Data Cleaning*

Description

Removes empty row/column or row/column with missing/ infinite values or zeros.

Usage

```
remove_NA(x, ...)

remove_Inf(x, ...)

remove_empty(x, ...)

remove_zero(x, ...)

## S4 method for signature 'matrix'
remove_NA(x, margin = 1)

## S4 method for signature 'matrix'
remove_Inf(x, margin = 1)

## S4 method for signature 'matrix'
remove_zero(x, margin = 1)

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

Arguments

x	A matrix, a data.frame or a *Matrix object.
...	Currently not used.
margin	An integer giving the subscript which the cleaning will be applied over (1 indicates rows, 2 indicates columns).

Author(s)

N. Frerebeau

See Also

Other utilities: replace()

Examples

```
## Create a count data matrix
X <- CountMatrix(sample(1:10, 25, TRUE), nrow = 5, ncol = 5)
k <- sample(1:25, 3, FALSE)

## Add zeros
X[k] <- 0L
## Remove row with zeros
remove_zero(X, margin = 1)
## Remove column with zeros
remove_zero(X, margin = 2)

## Add NA
X[k] <- NA
## Remove row with zeros
remove_NA(X, margin = 1)
## Remove column with zeros
remove_NA(X, margin = 2)
```

---

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 utilities: remove()

Examples

```
## Create a count data matrix
X <- CountMatrix(sample(1:10, 25, TRUE), nrow = 5, ncol = 5)
k <- sample(1:25, 3, FALSE)

## Add zeros
X[k] <- 0L
## Remove row with zeros
remove_zero(X, margin = 1)
## Remove column with zeros
remove_zero(X, margin = 2)

## Add NA
X[k] <- NA
## Remove row with zeros
remove_NA(X, margin = 1)
## Remove column with zeros
remove_NA(X, margin = 2)
```

`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

∗ **classes**
  coerce, 6
  CompositionMatrix-class, 9
  CountMatrix-class, 11
  IncidenceMatrix-class, 12
  OccurrenceMatrix-class, 17
  StratigraphicMatrix-class, 26
∗ **mutators**
  mutators, 13
  subset, 27
  summary, 28
∗ **predicates**
  predicate-matrix, 18
  predicate-numeric, 19
  predicate-scalar, 20
  predicate-trend, 20
  predicate-type, 21
  predicate-utils, 22
∗ **utilities**
  remove, 23
  replace, 24
∗ **validation methods**
  check-attribute, 2
  check-data, 3
  check-matrix, 4
  check-numeric, 5
  check-type, 6
  validate, 29
.CompositionMatrix *(CompositionMatrix-class)*, 9
.CountMatrix *(CountMatrix-class)*, 11
.IncidenceMatrix *(IncidenceMatrix-class)*, 12
.OccurrenceMatrix *(OccurrenceMatrix-class)*, 17
.StratigraphicMatrix *(StratigraphicMatrix-class)*, 26
[,AbundanceMatrix-method *(subset)*, 27
[<-,AbundanceMatrix-method *(subset)*, 27
[[<-,AbundanceMatrix-method *(subset)*, 27

AbundanceMatrix, *15*, *28*
as.integer(), *27*
as.vector, *10–12*
as_composition *(coerce)*, 6
as_composition(), *10*
as_composition,ANY-method *(coerce)*, 6
as_composition-method *(coerce)*, 6
as_count *(coerce)*, 6
as_count(), *11*
as_count,ANY-method *(coerce)*, 6
as_count-method *(coerce)*, 6
as_features *(coerce)*, 6
as_features,AbundanceMatrix-method *(coerce)*, 6
as_features-method *(coerce)*, 6
as_incidence *(coerce)*, 6
as_incidence(), *13*
as_incidence,ANY-method *(coerce)*, 6
as_incidence-method *(coerce)*, 6
as_long *(coerce)*, 6
as_long,AbundanceMatrix-method *(coerce)*, 6
as_long,matrix-method *(coerce)*, 6
as_long-method *(coerce)*, 6
as_occurrence *(coerce)*, 6
as_occurrence(), *17*
as_occurrence,ANY-method *(coerce)*, 6
as_occurrence-method *(coerce)*, 6
as_stratigraphy *(coerce)*, 6
as_stratigraphy(), *26*
as_stratigraphy,ANY-method *(coerce)*, 6
as_stratigraphy-method *(coerce)*, 6
assert_dimensions *(check-attribute)*, 2
assert_dimnames *(check-attribute)*, 2
assert_empty *(check-attribute)*, 2
assert_filled *(check-attribute)*, 2
assert_infinite *(check-data)*, 3
assert_length *(check-attribute)*, 2

assert_lengths (check-attribute), 2
assert_matrix (check-matrix), 4
assert_missing (check-data), 3
assert_names (check-attribute), 2
assert_numeric (check-numeric), 5
assert_relation (check-numeric), 5
assert_scalar (check-type), 6
assert_trend (check-numeric), 5
assert_type (check-type), 6
assert_unique (check-data), 3

character, *4–6*, *23*, *27*
check-attribute, 2
check-data, 3
check-matrix, 4
check-numeric, 5
check-type, 6
coerce, 6, *10, 11*, *13*, *17*, *26*
CompositionMatrix, *8*
CompositionMatrix (CompositionMatrix-class), 9
CompositionMatrix-class, 9
CountMatrix, *8*
CountMatrix (CountMatrix-class), 11
CountMatrix-class, 11

data.frame, *8*, *24, 25*
DataMatrix, *9–11*, *13*, *17*, *26*
dimnames, *10–12*

expression, *10–12*

factor, *7*

get (mutators), 13
get_dates (mutators), 13
get_dates,AbundanceMatrix-method (mutators), 13
get_dates-method (mutators), 13
get_groups (mutators), 13
get_groups,AbundanceMatrix-method (mutators), 13
get_groups-method (mutators), 13
get_samples (mutators), 13
get_samples,AbundanceMatrix-method (mutators), 13
get_samples-method (mutators), 13
get_taq (mutators), 13
get_taq,AbundanceMatrix-method (mutators), 13
get_taq-method (mutators), 13
get_terminus (mutators), 13
get_terminus,AbundanceMatrix-method (mutators), 13
get_terminus-method (mutators), 13
get_totals (mutators), 13
get_totals,AbundanceMatrix-method (mutators), 13
get_totals,OccurrenceMatrix-method (mutators), 13
get_totals-method (mutators), 13
get_tpq (mutators), 13
get_tpq,AbundanceMatrix-method (mutators), 13
get_tpq-method (mutators), 13

has_dates (mutators), 13
has_dates,AbundanceMatrix-method (mutators), 13
has_dates-method (mutators), 13
has_duplicates (predicate-utils), 22
has_groups (mutators), 13
has_groups,AbundanceMatrix-method (mutators), 13
has_groups-method (mutators), 13
has_infinite (predicate-utils), 22
has_length (predicate-utils), 22
has_missing (predicate-utils), 22
has_names (predicate-utils), 22
has_terminus (mutators), 13
has_terminus,AbundanceMatrix-method (mutators), 13
has_terminus-method (mutators), 13

IncidenceMatrix, *8*
IncidenceMatrix (IncidenceMatrix-class), 12
IncidenceMatrix-class, 12
infinite, *23, 24*
integer, *17*, *24*, *27*
is_atomic (predicate-type), 21
is_character (predicate-type), 21
is_constant (predicate-trend), 20
is_decreasing (predicate-trend), 20
is_double (predicate-type), 21
is_empty (predicate-utils), 22
is_error (predicate-type), 21
is_even (predicate-numeric), 19
is_greater (predicate-trend), 20

is_increasing (predicate-trend), 20
is_integer (predicate-type), 21
is_list (predicate-type), 21
is_logical (predicate-type), 21
is_lower (predicate-trend), 20
is_message (predicate-type), 21
is_negative (predicate-numeric), 19
is_numeric (predicate-type), 21
is_odd (predicate-numeric), 19
is_positive (predicate-numeric), 19
is_scalar_atomic (predicate-scalar), 20
is_scalar_character (predicate-scalar), 20
is_scalar_double (predicate-scalar), 20
is_scalar_integer (predicate-scalar), 20
is_scalar_list (predicate-scalar), 20
is_scalar_logical (predicate-scalar), 20
is_scalar_numeric (predicate-scalar), 20
is_scalar_vector (predicate-scalar), 20
is_square (predicate-matrix), 18
is_symmetric (predicate-matrix), 18
is_vector (predicate-type), 21
is_warning (predicate-type), 21
is_whole (predicate-numeric), 19
is_zero (predicate-numeric), 19

list, *8*, *16*
logical, *3*, *7*, *16*, *18–23*, *27*

matrix, *4*, *8*, *18*, *24, 25*
missing, *23, 24*
mutators, 13, *27*, *29*

numeric, *5*, *19*, *21*, *23*, *27*

OccurrenceMatrix, *8*
OccurrenceMatrix-class, 17

predicate-matrix, 18
predicate-numeric, 19
predicate-scalar, 20
predicate-trend, 20
predicate-type, 21
predicate-utils, 22

remove, 23, *25*
remove_empty (remove), 23
remove_empty,matrix-method (remove), 23
remove_empty-method (remove), 23
remove_Inf (remove), 23
remove_Inf,matrix-method (remove), 23
remove_Inf-method (remove), 23
remove_NA (remove), 23
remove_NA,matrix-method (remove), 23
remove_NA-method (remove), 23
remove_zero (remove), 23
remove_zero,matrix-method (remove), 23
remove_zero-method (remove), 23
replace, *24*, 24
replace_Inf (replace), 24
replace_Inf,matrix-method (replace), 24
replace_Inf-method (replace), 24
replace_NA (replace), 24
replace_NA,matrix-method (replace), 24
replace_NA-method (replace), 24
replace_zero (replace), 24
replace_zero,matrix-method (replace), 24
replace_zero-method (replace), 24

set (mutators), 13
set_dates,AbundanceMatrix,NULL-method (mutators), 13
set_dates,AbundanceMatrix,numeric-method (mutators), 13
set_dates-method (mutators), 13
set_dates<- (mutators), 13
set_dates<-,AbundanceMatrix,NULL-method (mutators), 13
set_dates<-,AbundanceMatrix,numeric-method (mutators), 13
set_groups,AbundanceMatrix-method (mutators), 13
set_groups-method (mutators), 13
set_groups<- (mutators), 13
set_groups<-,AbundanceMatrix-method (mutators), 13
set_samples,AbundanceMatrix-method (mutators), 13
set_samples-method (mutators), 13
set_samples<- (mutators), 13
set_samples<-,AbundanceMatrix-method (mutators), 13
set_taq,AbundanceMatrix,NULL-method (mutators), 13
set_taq,AbundanceMatrix,numeric-method (mutators), 13
set_taq-method (mutators), 13
set_taq<- (mutators), 13

set_taq<-,AbundanceMatrix,NULL-method
(mutators), 13
set_taq<-,AbundanceMatrix,numeric-method
(mutators), 13
set_terminus,AbundanceMatrix,list-method
(mutators), 13
set_terminus,AbundanceMatrix,NULL-method
(mutators), 13
set_terminus-method (mutators), 13
set_terminus<- (mutators), 13
set_terminus<-,AbundanceMatrix,list-method
(mutators), 13
set_terminus<-,AbundanceMatrix,NULL-method
(mutators), 13
set_totals,AbundanceMatrix-method
(mutators), 13
set_totals-method (mutators), 13
set_totals<- (mutators), 13
set_totals<-,AbundanceMatrix-method
(mutators), 13
set_tpq,AbundanceMatrix,NULL-method
(mutators), 13
set_tpq,AbundanceMatrix,numeric-method
(mutators), 13
set_tpq-method (mutators), 13
set_tpq<- (mutators), 13
set_tpq<-,AbundanceMatrix,NULL-method
(mutators), 13
set_tpq<-,AbundanceMatrix,numeric-method
(mutators), 13
StratigraphicMatrix, *8*
StratigraphicMatrix-class, 26
subset, *16*, 27, *29*
summary, *16*, *27*, 28
summary,AbundanceMatrix-method
(summary), 28

validate, *3–6*, 29
vector, *23*