Package 'arkhe'

May 14, 2021

Title Representation of Archaeological Data

Version 0.3.1

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

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

Depends R (>= 3.3)

Imports methods, stats

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' 'mutators.R' 'show.R' 'subset.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 2021-05-14 10:12:12 UTC

R topics documented:

`coerce` *Coerce*

Description

Coerce

Usage

```
as_long(from, ...)

as_count(from)

as_composition(from)

as_abundance(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_abundance(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.

Author(s)

N. Frerebeau

See Also

Other matrix: `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
B <- 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(B)
all(A1 == A2)

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

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

## Coerce to an S3 matrix or data.frame
```

```
X <- as.matrix(A1)
all(A0 == X)

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

## Collection of features
# set_dates(A1) <- matrix(sample(0:10, 20, TRUE), nrow = 10, ncol = 2)
# set_coordinates(A1) <- matrix(sample(0:10, 30, TRUE), nrow = 10, ncol = 3)
# as_features(A1)
```

`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.

Slots

`total` A `numeric` vector giving the absolute row sums.

Author(s)

N. Frerebeau

See Also

as_composition()

Other matrix: 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 matrix: 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 matrix: 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
```

`mutator` *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

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 'CompositionMatrix'
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 'CompositionMatrix'
set_totals(x) <- value
```

Arguments

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

Value

An object of the same sort as x with the new values assigned.

Author(s)

N. Frerebeau

See Also

Other mutator: subset()

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
```

`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 matrix: `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_missing(x, finite = FALSE)

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.
`finite`	A `logical` scalar: should non-`finite` values also be removed?
`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.
- `is_overlapping()` checks if two data ranges overlap at all.

Usage

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

is_increasing(x, na.rm = TRUE)

is_decreasing(x, na.rm = TRUE)
```

Arguments

`x`	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?

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 if a vector or list is empty.
- is_missing() checks if a vector or list contains missing values.
- is_named() checks if an object is named.
- is_uuid() checks if a string is a canonically formatted UUID that is version 1 through 5 and is the appropriate Variant as per RFC4122.

Usage

```
is_empty(x)

is_named(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-type

---

remove *Data Cleaning*

Description

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

Usage

```
remove_NA(x, ...)

remove_empty(x, ...)

remove_zero(x, ...)

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

## 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).
finite	A logical scalar: should non-finite values also be removed?

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

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

Usage

```
replace_NA(x, ...)

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

Arguments

x	A matrix, a data.frame or a *Matrix object.
...	Currently not used.
value	A possible value to replace missing 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 matrix: 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 method for signature 'CompositionMatrix'
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 mutator: mutator

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
```

Index