Package ‘tabula’

May 25, 2021

Title Analysis, Seriation and Visualization of Archaeological Count Data

Version 1.6.1

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

Description An easy way to examine archaeological count data. This package provides a convenient and reproducible toolkit for relative dating by matrix seriation (reciprocal ranking, CA-based seriation). It also provides several tests and measures of diversity: heterogeneity and evenness (Brillouin, Shannon, Simpson, etc.), richness and rarefaction (Chao1, Chao2, ACE, ICE, etc.), turnover and similarity (Brainerd-Robinson, etc.). The package make it easy to visualize count data and statistical thresholds: rank vs abundance plots, heatmaps, Ford (1962) and Bertin (1977) diagrams.

License GPL-3


BugReports https://github.com/tesselle/tabula/issuesa

Depends R (>= 3.3)

Imports arkhe (>= 0.3.0), dimensio, ggplot2, grDevices, methods, rlang, stats, utils

Suggests covr, folio, khroma, knitr, magrittr, rmarkdown, testthat (>= 3.0.0), vdiffr

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.1.1

Collate 'AllClasses.R' 'AllGenerics.R' 'coerce.R' 'deprecate.R' 'ggplot2.R' 'index_diversity.R' 'index_heterogeneity.R' 'index_rarefaction.R' 'index_richness.R' 'index_similarity.R' 'index_turnover.R' 'mutators.R' 'permute.R' 'plot_bertin.R' 'plot_diversity.R' 'plot_ford.R' 'plot_heatmap.R' 'plot_rank.R'
## Description

An S4 class to represent a diversity measure.
heterogeneity-index

Slots

- names: A character vector giving the sample names.
- values: A numeric vector giving the diversity index values.
- size: A integer vector giving the sample sizes.
- simulation: A four columns numeric matrix giving the diversity measures for the simulated assemblage (sample size, mean estimate, lower and upper boundaries of the confidence interval).
- method: A character string specifying the method used.

Subset

In the code snippets below, x is a DiversityIndex object.

x[[i]]: Extracts information from a slot selected by subscript i. i is a length-one character vector. Returns the corresponding slot values.

Author(s)

N. Frerebeau

See Also

Other class: PermutationOrder-class, RefineCA-class

heterogeneity-index  Heterogeneity and Evenness

Description

- index_heterogeneity() returns an heterogeneity or dominance index.
- index_evenness() returns an evenness measure.
- bootstrap_*() and jackknife_*() perform bootstrap/jackknife resampling.

Usage

index_heterogeneity(object, ...)
simulate_heterogeneity(object, ...)
bootstrap_heterogeneity(object, ...)
jackknife_heterogeneity(object, ...)
index_evenness(object, ...)
simulate_evenness(object, ...)

bootstrap_evenness(object, ...)

jackknife_evenness(object, ...)

## S4 method for signature 'CountMatrix'
index_heterogeneity(
  object,
  method = c("berger", "brillouin", "mcintosh", "shannon", "simpson"),
  ...
)

## S4 method for signature 'CountMatrix'
simulate_heterogeneity(
  object,
  method = c("berger", "brillouin", "mcintosh", "shannon", "simpson"),
  quantiles = TRUE,
  level = 0.8,
  step = 1,
  n = 1000,
  progress = getOption("tabula.progress"),
  ...
)

## S4 method for signature 'CountMatrix'
bootstrap_heterogeneity(
  object,
  method = c("berger", "brillouin", "mcintosh", "shannon", "simpson"),
  probs = c(0.05, 0.95),
  n = 1000,
  ...
)

## S4 method for signature 'CountMatrix'
jackknife_heterogeneity(
  object,
  method = c("berger", "brillouin", "mcintosh", "shannon", "simpson"),
  ...
)

## S4 method for signature 'CountMatrix'
index_evenness(
  object,
  method = c("shannon", "brillouin", "mcintosh", "simpson"),
  ...
)

## S4 method for signature 'CountMatrix'
simulate_evenness(
  object,
  method = c("shannon", "brillouin", "mcintosh", "simpson"),
  quantiles = TRUE,
  level = 0.8,
  step = 1,
  n = 1000,
  progress = getOption("tabula.progress"),
  ...
)

## S4 method for signature 'CountMatrix'

bootstrap_evenness(
  object,
  method = c("berger", "brillouin", "mcintosh", "shannon", "simpson"),
  probs = c(0.05, 0.95),
  n = 1000,
  ...
)

## S4 method for signature 'CountMatrix'

jackknife_evenness(
  object,
  method = c("berger", "brillouin", "mcintosh", "shannon", "simpson"),
  ...
)

Arguments

- **object** A \( m \times p \) matrix of count data (typically a \texttt{CountMatrix} object).
- ... Further arguments to be passed to internal methods.
- **method** A \texttt{character} string specifying the index to be computed (see details). Any unambiguous substring can be given.
- **quantiles** A \texttt{logical} scalar: should sample quantiles be used as confidence interval? If TRUE (the default), sample quantiles are used as described in Kintigh (1989), else quantiles of the normal distribution are used.
- **level** A length-one \texttt{numeric} vector giving the confidence level.
- **step** A non-negative \texttt{integer} giving the increment of the sample size. Only used if simulate is TRUE.
- **n** A non-negative \texttt{integer} giving the number of bootstrap replications.
- **progress** A \texttt{logical} scalar: should a progress bar be displayed?
- **probs** A \texttt{numeric} vector of probabilities with values in \([0, 1]\) (see \texttt{stats::quantile()}).

Details

Diversity measurement assumes that all individuals in a specific taxa are equivalent and that all types are equally different from each other (Peet 1974). A measure of diversity can be achieved by...
using indices built on the relative abundance of taxa. These indices (sometimes referred to as non-parametric indices) benefit from not making assumptions about the underlying distribution of taxa abundance: they only take relative abundances of the species that are present and species richness into account. Peet (1974) refers to them as indices of heterogeneity.

Diversity indices focus on one aspect of the taxa abundance and emphasize either richness (weighting towards uncommon taxa) or dominance (weighting towards abundant taxa; Magurran 1988).

Evenness is a measure of how evenly individuals are distributed across the sample.

The following heterogeneity index and corresponding evenness measures are available (see Magurran 1988 for details):

- **berger** Berger-Parker dominance index. The Berger-Parker index expresses the proportional importance of the most abundant type. This metric is highly biased by sample size and richness, moreover it does not make use of all the information available from sample.

- **brillouin** Brillouin diversity index. The Brillouin index describes a known collection: it does not assume random sampling in an infinite population. Pielou (1975) and Laxton (1978) argues for the use of the Brillouin index in all circumstances, especially in preference to the Shannon index.

- **mcintosh** McIntosh dominance index. The McIntosh index expresses the heterogeneity of a sample in geometric terms. It describes the sample as a point of a $S$-dimensional hypervolume and uses the Euclidean distance of this point from the origin.

- **shannon** Shannon-Wiener diversity index. The Shannon index assumes that individuals are randomly sampled from an infinite population and that all taxa are represented in the sample (it does not reflect the sample size). The main source of error arises from the failure to include all taxa in the sample: this error increases as the proportion of species discovered in the sample declines (Peet 1974, Magurran 1988). The maximum likelihood estimator (MLE) is used for the relative abundance, this is known to be negatively biased by sample size.

- **simpson** Simpson dominance index for finite sample. The Simpson index expresses the probability that two individuals randomly picked from a finite sample belong to two different types. It can be interpreted as the weighted mean of the proportional abundances. This metric is a true probability value, it ranges from 0 (perfectly uneven) to 1 (perfectly even).

The berger, mcintosh and simpson methods return a dominance index, not the reciprocal or inverse form usually adopted, so that an increase in the value of the index accompanies a decrease in diversity.

**Value**

- `index_heterogeneity()`, `index_evenness()` and `simulate_evenness()` return a `DiversityIndex` object.
- `bootstrap_*()` and `jackknife_*()` return a `data.frame`.

**Note**

Ramanujan approximation is used for $x!$ computation if $x > 170$.

**Author(s)**

N. Frerebeau
References


See Also

`plot_diversity()`, `similarity()`, `turnover()`

Other diversity: `richness-index`, `similarity()`, `turnover-index`

Examples

```r
## Coerce dataset to a count matrix
data("chevelon", package = "folio")
chevelon <- as_count(chevelon)

## Shannon diversity index
(index_h <- index_heterogeneity(chevelon, method = "shannon"))
(index_e <- index_evenness(chevelon, method = "shannon"))

## Bootstrap resampling
(boot_h <- bootstrap_heterogeneity(chevelon, method = "shannon"))

## Jackknife resampling
(jack_h <- jackknife_heterogeneity(chevelon, method = "shannon"))
```
Description

Independance

Usage

eppm(object, ...)

pvi(object, ...)

## S4 method for signature 'CountMatrix'
eppm(object)

## S4 method for signature 'CountMatrix'
pvi(object)

Arguments

object A CountMatrix object.

... Currently not used.

Details

Computes for each cell of a numeric matrix one of the following statistic.

Value

A numeric matrix.

EPPM

This positive difference from the column mean percentage (in French "écart positif au pourcentage moyen", EPPM) represents a deviation from the situation of statistical independence. As independence can be interpreted as the absence of relationships between types and the chronological order of the assemblages, EPPM is a useful graphical tool to explore significance of relationship between rows and columns related to seriation (Desachy 2004).

PVI

PVI is calculated for each cell as the percentage to the column theoretical independence value: PVI greater than 1 represent positive deviations from the independence, whereas PVI smaller than 1 represent negative deviations (Desachy 2004).

The PVI matrix allows to explore deviations from independence (an intuitive graphical approach to $\chi^2$), in such a way that a high-contrast matrix has quite significant deviations, with a low risk of being due to randomness (Desachy 2004).
### mutator

**Description**

Getters and setters to extract or replace parts of an object.

**Usage**

```r
get_index(x)
get_method(x)
get_order(x)
```

```r
## S4 method for signature 'HeterogeneityIndex'
get_index(x)
```

```r
## S4 method for signature 'EvennessIndex'
get_index(x)
```
## S4 method for signature 'RichnessIndex'
get_index(x)

## S4 method for signature 'DiversityIndex'
get_method(x)

### Arguments

- **x**: An object from which to get or set element(s).

### Value

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

### Author(s)

N. Frerebeau

### See Also

Other mutator: `subset()`

---

**PermutationOrder-class**

*Permutation Order*

### Description

An S4 class to represent a permutation order.

### Slots

- **rows**: An integer vector giving the rows permutation.
- **columns**: An integer vector giving the columns permutation.
- **method**: A character string indicating the seriation method used.

### Subset

In the code snippets below, `x` is a `PermutationOrder` object.

- `x[[i]]`: Extracts information from a slot selected by subscript `i`. `i` is a length-one character vector. Returns the corresponding slot values.

### Author(s)

N. Frerebeau

### See Also

Other class: `DiversityIndex, RefineCA-class`
**Description**

Plots a Bertin, Ford (battleship curve) or Dice-Leraas diagram.

**Usage**

```r
plot_bertin(object, ...)

plot_ford(object, ...)
```

```r
## S4 method for signature 'matrix'
plot_bertin(object, threshold = NULL, scale = NULL)

## S4 method for signature 'matrix'
plot_ford(object)

## S4 method for signature 'CountMatrix'
plot_ford(object, EPPM = FALSE)
```

**Arguments**

- `object`: An abundance matrix to be plotted.
- `...`: Currently not used.
- `threshold`: A function that takes a numeric vector as argument and returns a numeric threshold value (see below). If NULL (the default), no threshold is computed.
- `scale`: A function used to scale each variable, that takes a numeric vector as argument and returns a numeric vector. If NULL (the default), no scaling is performed.
- `EPPM`: A logical scalar: should the EPPM be drawn (see below)?

**Details**

If EPPM is TRUE and if a relative abundance is greater than the mean percentage of the type, the exceeding part is highlighted.

**Value**

A `ggplot2::ggplot` object.

**Bertin Matrix**

As de Falguerolles *et al.* (1997) points out: "In abstract terms, a Bertin matrix is a matrix of displays. ... To fix ideas, think of a data matrix, variable by case, with real valued variables. For each variable, draw a bar chart of variable value by case. Highlight all bars representing a value above some sample threshold for that variable."
EPPM

This positive difference from the column mean percentage (in French "écart positif au pourcentage moyen", EPPM) represents a deviation from the situation of statistical independence. As independence can be interpreted as the absence of relationships between types and the chronological order of the assemblages, EPPM is a useful graphical tool to explore significance of relationship between rows and columns related to seriation (Desachy 2004).

Author(s)

N. Frerebeau

References


See Also

eppm()

Other plot: plot_diversity, plot_line, plot_matrix, plot_spot()

Examples

```r
## Abundance data
## Coerce dataset to a count matrix
data("mississippi", package = "folio")
counts1 <- as_count(mississippi)

## Plot a Bertin diagram...
## ...without threshold
plot_bertin(counts1)

## ...with variables scaled to 0-1 and the variable mean as threshold
scale_01 <- function(x) (x - min(x)) / (max(x) - min(x))
plot_bertin(counts1, threshold = mean, scale = scale_01)

## Abundance data
## Coerce dataset to a count matrix (data from Desachy 2004)
data("compiegne", package = "folio")
counts2 <- as_count(compiegne)

## Plot a Ford diagram...
```
plot_diversity

## ...without threshold
plot_ford(counts2)
## ...with EPPM
plot_ford(counts2, EPPM = TRUE)

---

plot_diversity       Diversity Plot

### Description
Diversity Plot

### Usage

```r
## S4 method for signature 'DiversityIndex,missing'
plot(x)
```

### Arguments

- `x` A `DiversityIndex` object to be plotted.

### Author(s)
N. Frerebeau

### See Also

- `index_heterogeneity()`, `index_evenness()`, `index_richness()`

Other plot: `plot_bar`, `plot_line`, `plot_matrix`, `plot_spot()`

### Examples

```r
## Coerce data to a count matrix
data("chevelon", package = "folio")
chevelon <- as_count(chevelon)

## Assemblage diversity size comparison
## Warning: this may take a few seconds!
sim_evenness <- simulate_evenness(chevelon, method = "shannon")
plot(sim_evenness)

sim_richness <- simulate_richness(chevelon, method = "none")
plot(sim_richness)
```
plot_line

Line Plot

Description

Plots a rank vs relative abundance diagram.

Usage

plot_rank(object, ...)

## S4 method for signature 'matrix'
plot_rank(object, log = NULL, facet = FALSE)

Arguments

object

An abundance matrix to be plotted.

...

Further arguments to be passed to internal methods.

log

A character string which contains "x" if the x axis is to be logarithmic, "y" if the y axis is to be logarithmic and "xy" or "yx" if both axes are to be logarithmic (base 10).

facet

A logical scalar: should a matrix of panels defined by case/sample be drawn?

Value

A ggplot2::ggplot object.

Author(s)

N. Frerebeau

References


See Also

Other plot: plot_bar, plot_diversity, plot_matrix, plot_spot()

Examples

## Abundance matrix
## Coerce datasets to a count matrix (data from Desachy 2004)
data("compiegne", package = "folio")
counts <- as_count(compiegne)

## Plot rank vs abundance
plot_matrix

plot_rank(counts)
plot_rank(counts, facet = TRUE)

plot_matrix  Heatmap

Description
Plots a heatmap.

Usage
plot_heatmap(object, ...)

## S4 method for signature 'matrix'
plot_heatmap(object)

Arguments

object  An object to be plotted.
...
Further arguments to be passed to internal methods.

Value
A ggplot2::ggplot object.

Author(s)
N. Frerebeau

References

See Also
pvi()
Other plot: plot_bar, plot_diversity, plot_line, plot_spot()
## Examples

```r
## Abundance data (data from Desachy 2004)
data("compiegne", package = "folio")
## Coerce dataset to absolute frequencies
counts <- as_count(compiegne)
## Coerce dataset to relative frequencies
freq <- as_composition(compiegne)

## Plot matrix diagram...
plot_heatmap(counts)
plot_heatmap(freq)

## Presence/absence data
inc <- sample(0:1, size = 100, replace = TRUE)
bin <- IncidenceMatrix(data = inc, nrow = 10, ncol = 10)

plot_heatmap(bin) +
  ggplot2::scale_fill_manual(values = c("TRUE" = "black", "FALSE" = "white"))
```

---

### plot_spot

**Spot Plot**

Plots a spot matrix.

#### Usage

```r
plot_spot(object, ...)  
## S4 method for signature 'matrix'
plot_spot(object, threshold = NULL, diag = TRUE, upper = TRUE, ...)  
## S4 method for signature 'dist'
plot_spot(object, diag = FALSE, upper = FALSE, ...)  
## S4 method for signature 'OccurrenceMatrix'
plot_spot(object, diag = FALSE, upper = FALSE, ...)  
```

#### Arguments

- `object`: An abundance matrix to be plotted.
- `...`: Extra parameters to be passed to `threshold`.
- `threshold`: A `function` that takes a numeric vector as argument and returns a numeric threshold value. If `NULL` (the default), no threshold is computed.
- `diag`: A `logical` scalar indicating whether the diagonal of the matrix should be plotted.
- `upper`: A `logical` scalar indicating whether the upper triangle of the matrix should be plotted.
Details

The spot matrix can be considered as a variant of the Bertin diagram where the data are first transformed to relative frequencies.

Value

A `ggplot2::ggplot` object.

Note

Adapted from Dan Gopstein’s original idea.

Author(s)

N. Frerebeau

See Also

Other plot: `plot_bar`, `plot_diversity`, `plot_line`, `plot_matrix`

Examples

```r
## Plot spot diagram of count data...
data("mississippi", package = "folio")
counts <- as_count(mississippi)

### ...without threshold
plot_spot(counts)
### ...with the column means as threshold
plot_spot(counts, threshold = mean)
### ...with the column medians as threshold
plot_spot(counts, threshold = median)
```

RefineCA-class

Partial Bootstrap CA

Description

An S4 class to store partial bootstrap correspondence analysis results.

Slots

- `row_chull` A three columns numeric matrix giving the vertices coordinates \((x, y)\) of the samples convex hull and a identifier \((id)\) to link each row to a sample.
- `row_lengths` A named numeric vector giving the convex hull maximum dimension length of samples.
- `row_keep` An integer vector giving the subscript of the samples to be kept.
column_chull A three columns numeric matrix giving the vertices coordinates \((x, y)\) of the variables convex hull and a identifier \((id)\) to link each row to a variable.

column_lengths A numeric vector giving the convex hull maximum dimension length of variables.

column_keep An integer vector giving the subscript of the variables to be kept.

cutoff A length-two numeric vector giving the cutoff value for samples and variables selection, respectively.

Subset

In the code snippets below, \(x\) is a RefineCA object.

\[x[[i]]\] Extracts information from a slot selected by subscript \(i\). \(i\) is a length-one character vector. Returns the corresponding slot values.

Author(s)

N. Frerebeau

See Also

dimensio::CA

Other class: DiversityIndex, PermutationOrder-class

richness-index Richness and Rarefaction

Description

- `index_richness()` returns sample richness.
- `index_composition()` returns asymptotic species richness.
- `rarefaction()` returns Hurlbert’s unbiased estimate of Sander’s rarefaction.
- `bootstrap_*()` and `jackknife_*()` perform bootstrap/jackknife resampling.

Usage

`index_richness(object, ...)`

`simulate_richness(object, ...)`

`bootstrap_richness(object, ...)`

`jackknife_richness(object, ...)`

`index_composition(object, ...)`
rarefaction(object, ...)

## S4 method for signature 'CountMatrix'
rarefaction(object, sample, method = c("hurlbert"), simplify = TRUE, ...)

## S4 method for signature 'CountMatrix'
index_richness(object, method = c("none", "margalef", "menhinick"), ...)

## S4 method for signature 'CountMatrix'
simulate_richness(
  object,
  method = c("none", "margalef", "menhinick"),
  quantiles = TRUE,
  level = 0.8,
  step = 1,
  n = 1000,
  progress = getOption("tabula.progress"),
  ...
)

## S4 method for signature 'CountMatrix'
bootstrap_richness(
  object,
  method = c("none", "margalef", "menhinick"),
  probs = c(0.05, 0.95),
  n = 1000,
  ...
)

## S4 method for signature 'CountMatrix'
jackknife_richness(object, method = c("none", "margalef", "menhinick"), ...)

## S4 method for signature 'CountMatrix'
index_composition(
  object,
  method = c("chao1", "ace"),
  unbiased = FALSE,
  improved = FALSE,
  k = 10
)

## S4 method for signature 'IncidenceMatrix'
index_composition(
  object,
  method = c("chao2", "ice"),
  unbiased = FALSE,
  improved = FALSE,
  k = 10
)
Arguments

object

A \( m \times p \) matrix of count data.

... Further arguments to be passed to internal methods.

sample

A length-one numeric vector giving the sub-sample size.

method

A character string or vector of strings specifying the index to be computed (see details). Any unambiguous substring can be given.

simplify

A logical scalar: should the result be simplified to a matrix? The default value, FALSE, returns a list.

quantiles

A logical scalar: should sample quantiles be used as confidence interval? If TRUE (the default), sample quantiles are used as described in Kintigh (1989), else quantiles of the normal distribution are used.

level

A length-one numeric vector giving the confidence level.

step

A non-negative integer giving the increment of the sample size. Only used if simulate is TRUE.

n

A non-negative integer giving the number of bootstrap replications.

progress

A logical scalar: should a progress bar be displayed?

probs

A numeric vector of probabilities with values in \([0, 1]\) (see stats::quantile()).

unbiased

A logical scalar. Should the bias-corrected estimator be used? Only used with "chao1" or "chao2" (improved) estimator.

improved

A logical scalar. Should the improved estimator be used? Only used with "chao1" or "chao2".

k

A length-one numeric vector giving the threshold between rare/infrequent and abundant/frequent species. Only used if method is "ace" or "ice".

Details

The number of different taxa, provides an instantly comprehensible expression of diversity. While the number of taxa within a sample is easy to ascertain, as a term, it makes little sense: some taxa may not have been seen, or there may not be a fixed number of taxa (e.g. in an open system; Peet 1974). As an alternative, richness \( (S) \) can be used for the concept of taxa number (McIntosh 1967).

It is not always possible to ensure that all sample sizes are equal and the number of different taxa increases with sample size and sampling effort (Magurran 1988). Then, rarefaction \( (E(S)) \) is the number of taxa expected if all samples were of a standard size (i.e. taxa per fixed number of individuals). Rarefaction assumes that imbalances between taxa are due to sampling and not to differences in actual abundances.

The following richness measures are available for count data:

margalef Margalef richness index.

menhinick Menhinick richness index.

none Returns the number of observed taxa/types.
Value

- `index_richness()`, `simulate_richness()` and `index_composition()` return a `DiversityIndex` object.
- `bootstrap_*()` and `jackknife_*()` return a `data.frame`.

If `simplify` is `FALSE`, then `rarefaction()` returns a `list` (default), else return a `matrix`.

Asymptotic Species Richness

The following measures are available for count data:

ace Abundance-based Coverage Estimator.
chaol (improved/unbiased) Chao1 estimator.

The following measures are available for replicated incidence data:

ice Incidence-based Coverage Estimator.
chaol2 (improved/unbiased) Chao2 estimator.

Author(s)

N. Frerebeau

References


See Also

- `plot_diversity()`

Other diversity: `heterogeneity-index`, `similarity()`, `turnover-index`

Examples

```r
## Richness
## Margalef and Menhinick index
## Data from Magurran 1988, p. 128-129
trap <- CountMatrix(data = c(9, 3, 0, 4, 2, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 2, 0, 5, 3, 0),
                     nrow = 2, byrow = TRUE, dimnames = list(c("A", "B"), NULL))
index_richness(trap, method = "margalef") # 2.55 1.88
index_richness(trap, method = "menhinick") # 1.95 1.66

## Asymptotic species richness
## Chao1-type estimators
## Data from Chao & Chiu 2016
brazil <- CountMatrix(
                     data = rep(x = c(1:21, 23, 25, 27, 28, 30, 32, 34:37, 41,
                                    45, 46, 49, 52, 89, 110, 123, 140),
                                    times = c(113, 50, 39, 29, 15, 11, 13, 5, 6, 6, 3, 4,
                                            3, 5, 2, 5, 2, 2, 2, 1, 1, 1, 1, 1, 1, 0, 2, 1, 1, 1, 0, 1, 0, 0)),
                     nrow = 1, byrow = TRUE)
index_composition(brazil, method = c("chao1"), unbiased = FALSE) # 461.625
index_composition(brazil, method = c("ace"), k = 10) # 445.822

## Rarefaction
rarefaction(trap, sample = 13) # 6.56
```

### Description

- `seriate_*()` computes a permutation order for rows and/or columns.
- `permute()` rearranges a data matrix according to a permutation order.
- `get_order()` returns the seriation order for rows and columns.
Usage

seriate_average(object, ...)

seriate_rank(object, ...)

permute(object, order, ...)

refine_seriation(object, ...)

### S4 method for signature 'PermutationOrder'
get_order(x)

### S4 method for signature 'CountMatrix,PermutationOrder'
permute(object, order)

### S4 method for signature 'IncidenceMatrix,PermutationOrder'
permute(object, order)

### S4 method for signature 'CA'
refine_seriation(
  object,
  cutoff,
  n = 1000,
  axes = c(1, 2),
  progress = getOption("tabula.progress"),
  ...
)

### S4 method for signature 'CountMatrix'
seriate_average(object, margin = c(1, 2), axes = 1, ...)

### S4 method for signature 'IncidenceMatrix'
seriate_average(object, margin = c(1, 2), axes = 1, ...)

### S4 method for signature 'CountMatrix'
seriate_rank(object, EPPM = FALSE, margin = c(1, 2), stop = 100)

### S4 method for signature 'IncidenceMatrix'
seriate_rank(object, margin = c(1, 2), stop = 100)

Arguments

object, x

An \(m \times p\) data matrix (typically an object of class \texttt{CountMatrix} or \texttt{IncidenceMatrix}.

... Further arguments to be passed to internal methods.

order A \texttt{PermutationOrder} object giving the permutation order for rows and columns.

cutoff A function that takes a numeric vector as argument and returns a single numeric value (see below).
A non-negative integer giving the number of bootstrap replications.

An integer vector giving the subscripts of the CA axes to be used.

A logical scalar: should a progress bar be displayed?

A numeric vector giving the subscripts which the rearrangement will be applied over: 1 indicates rows, 2 indicates columns, c(1,2) indicates rows then columns, c(2,1) indicates columns then rows.

A logical scalar: should the seriation be computed on EPPM instead of raw data?

An integer giving the stopping rule (i.e. maximum number of iterations) to avoid infinite loop.

• seriate_*() returns a PermutationOrder object.
• refine_seriation() returns a RefineCA object.
• permute() returns either a permuted CountMatrix or an IncidenceMatrix (the same as object).

Seriation

The matrix seriation problem in archaeology is based on three conditions and two assumptions, which Dunell (1970) summarizes as follows.

The homogeneity conditions state that all the groups included in a seriation must:

1. Be of comparable duration.
2. Belong to the same cultural tradition.
3. Come from the same local area.

The mathematical assumptions state that the distribution of any historical or temporal class:

1. Is continuous through time.
2. Exhibits the form of a unimodal curve.

Theses assumptions create a distributional model and ordering is accomplished by arranging the matrix so that the class distributions approximate the required pattern. The resulting order is inferred to be chronological.

The following seriation methods are available:

seriate_average() Correspondence analysis-based seriation (average ranking). Correspondence analysis (CA) is an effective method for the seriation of archaeological assemblages. The order of the rows and columns is given by the coordinates along one dimension of the CA space, assumed to account for temporal variation. The direction of temporal change within the correspondence analysis space is arbitrary: additional information is needed to determine the actual order in time.

seriate_rank() Reciprocal ranking seriation. These procedures iteratively rearrange rows and/or columns according to their weighted rank in the data matrix until convergence. Note that this procedure could enter into an infinite loop. If no convergence is reached before the maximum number of iterations, it stops with a warning.
Correspondence Analysis

refine_seriation() allows to identify samples that are subject to sampling error or samples that have underlying structural relationships and might be influencing the ordering along the CA space. This relies on a partial bootstrap approach to CA-based seriation where each sample is replicated $n$ times. The maximum dimension length of the convex hull around the sample point cloud allows to remove samples for a given cutoff value.

According to Peebles and Schachner (2012), "[this] point removal procedure [results in] a reduced dataset where the position of individuals within the CA are highly stable and which produces an ordering consistent with the assumptions of frequency seriation."

Note

Refining method can lead to much longer execution times and larger output objects.

Author(s)

N. Frerebeau

References


See Also

dimensio::ca()

Examples

```r
## Replicates Desachy 2004 results
## Coerce dataset to abundance matrix
data("compiegne", package = "folio")
compiegne_count <- as_count(compiegne)

## Get seriation order for columns on EPPM using the reciprocal averaging method
(compiegne_indices <- seriate_rank(compiegne_count, EPPM = TRUE, margin = 2))

## Permute columns
compiegne_new <- permute(compiegne_count, compiegne_indices)
```
## Plot new matrix
plot_ford(compiegne_new, EPPM = FALSE)

## See the vignette:
## Not run:
utils::vignette("seriation")
## End(Not run)

---

### similarity

**Description**

Similarity

**Usage**

`similarity(object, ...)`

```r
## S4 method for signature 'CountMatrix'
similarity(
  object,
  method = c("brainerd", "bray", "jaccard", "morisita", "sorenson", "binomial"),
  ...
)
```

```r
## S4 method for signature 'IncidenceMatrix'
similarity(object, method = c("jaccard", "sorenson"), ...)
```

**Arguments**

- `object`  
  A \( m \times p \) matrix of count data.

- `...`  
  Further arguments to be passed to internal methods.

- `method`  
  A **character** string specifying the method to be used (see details). Any unambiguous substring can be given.

**Details**

\( \beta \)-diversity can be measured by addressing *similarity* between pairs of samples/cases (Brainerd-Robinson, Jaccard, Morisita-Horn and Sorenson indices). Similarity between pairs of taxa/types can be measured by assessing the degree of co-occurrence (binomial co-occurrence).

Jaccard, Morisita-Horn and Sorenson indices provide a scale of similarity from 0-1 where 1 is perfect similarity and 0 is no similarity. The Brainerd-Robinson index is scaled between 0 and 200. The Binomial co-occurrence assessment approximates a Z-score.
binomial  Binomial co-occurrence assessment. This assesses the degree of co-occurrence between taxa/types within a dataset. The strongest associations are shown by large positive numbers, the strongest segregations by large negative numbers.

brainerd  Brainerd-Robinson quantitative index. This is a city-block metric of similarity between pairs of samples/cases.

bray    Sorenson quantitative index (Bray and Curtis modified version of the Sorenson index).

jaccard Jaccard qualitative index.

morisita Morisita-Horn quantitative index.

sorenson Sorenson qualitative index.

Value

similarity() returns a stats::dist object.

Author(s)

N. Frerebeau

References


See Also

Other diversity: heterogeneity-index, richness-index, turnover-index

Examples

## Data from Huntley 2008

```r
cermics <- CountMatrix(
  data = c(16, 9, 3, 0, 1, 13, 3, 2, 0, 0, 9, 5, 2, 5, 0, 14, 12, 3, 0, 0, 0, 26, 4, 0, 0, 1, 26, 4, 0, 0, 0, 11, 3, 13, 0, 0, 0, 17, 0, 16),
```
subset

Extract or Replace Parts of an Object

Description

Operators acting on objects to extract or replace parts.

Usage

## S4 method for signature 'DiversityIndex,ANY,missing'
x[[i]]

## S4 method for signature 'RefineCA,ANY,missing'
x[[i]]

## S4 method for signature 'PermutationOrder,ANY,missing'
x[[i]]
**test_diversity**

**Arguments**

- `x`  
  An object from which to extract element(s) or in which to replace element(s).

- `i`  
  A character string specifying elements to extract. Any unambiguous substring can be given (see details).

**Value**

A subsetted object.

**Author(s)**

N. Frerebeau

**See Also**

Other mutator: `mutator`

---

**test_diversity**  
*Diversity Test*

---

**Description**

Compares Shannon diversity between samples.

**Usage**

```
test_diversity(object, ...)  
```  
## S4 method for signature 'CountMatrix'

```
test_diversity(object, adjust = "holm", ...)  
```  

**Arguments**

- `object`  
  A \( m \times p \) matrix of count data.

- `...`  
  Further arguments to be passed to internal methods.

- `adjust`  
  A character string specifying the method for adjusting \( p \) values (see `stats::p.adjust()`).

**Details**

This test produces two sided pairwise comparisons: it returns a matrix of adjusted \( p \) values.

**Value**

A numeric matrix.

**Author(s)**

N. Frerebeau
References


See Also

Other statistics: independance

Examples

```r
## Shannon diversity test
data("merzbach", package = "folio")
merzbach_count <- as_count(merzbach)
div <- test_diversity(merzbach_count)
```

---

**Description**

Returns the degree of turnover in taxa composition along a gradient or transect.

**Usage**

```r
turnover(object, ...)
```

**Arguments**

- **object**: A $m \times p$ matrix of count data.
- **...**: Further arguments to be passed to internal methods.
- **method**: A character string specifying the method to be used (see details). Any unambiguous substring can be given.
- **simplify**: A logical scalar: should the result be simplified to a matrix?
Details

The following methods can be used to ascertain the degree of turnover in taxa composition along a gradient ($\beta$-diversity) on qualitative (presence/absence) data. This assumes that the order of the matrix rows (from 1 to $n$) follows the progression along the gradient/transect.

**whittaker** Whittaker measure.

**cody** Cody measure.

**routledge1** Routledge first measure.

**routledge2** Routledge second measure.

**routledge3** Routledge third measure. This is the exponential form of the second measure.

**wilson** Wilson measure.

Value

If simplify is FALSE, returns a list (default), else returns a matrix.

Author(s)

N. Frerebeau

References


See Also

Other diversity: `heterogeneity-index`, `richness-index`, `similarity()`

Examples

```r
## Data from Magurran 1988, p. 162
trees <- IncidenceMatrix(
  data = c(1, 1, 1, 0, 0, 0,
           1, 1, 1, 1, 1, 1,
           0, 0, 1, 0, 1, 0,
           0, 0, 0, 1, 1, 1,
           0, 0, 0, 0, 1, 1,
           0, 0, 0, 1, 0, 1),
  nrow = 6, byrow = FALSE,
  dimnames = list(c("1", "2", "3", "4", "5", "6"),
```
c("Birch", "Oak", "Rowan", "Beech", "Hazel", "Holly")
)

## Whittaker's measure
turnover(trees, "whittaker") # 1

## Cody's measure
turnover(trees, "cody") # 3

## Routledge's measures
turnover(trees, method = c("routledge1", "routledge2", "routledge3"),
  simplify = TRUE) ## 0.29 0.56 1.75

## Wilson and Shmida's measure
turnover(trees, "wilson") # 1
Index

* class
  DiversityIndex, 2
  PermutationOrder-class, 10
  RefineCA-class, 17

* diversity
  heterogeneity-index, 3
  richness-index, 18
  similarity, 26
  turnover-index, 30

* mutator
  mutator, 9
  subset, 28

* plot
  plot_bar, 11
  plot_diversity, 13
  plot_line, 14
  plot_matrix, 15
  plot_spot, 16

* seriation
  seriation, 22

* statistics
  independance, 8
  test_diversity, 29

Bertin diagram, 17
bootstrap_evenness
  (heterogeneity-index), 3
bootstrap_evenness,CountMatrix-method
  (heterogeneity-index), 3
bootstrap_evenness-method
  (heterogeneity-index), 3
bootstrap_heterogeneity
  (heterogeneity-index), 3
bootstrap_heterogeneity,CountMatrix-method
  (heterogeneity-index), 3
bootstrap_heterogeneity-method
  (heterogeneity-index), 3
bootstrap_richness (richness-index), 18
bootstrap_richness,CountMatrix-method
  (richness-index), 18
bootstrap_richness-method
  (richness-index), 18

character, 3, 5, 10, 14, 18, 20, 26, 29, 30
CompositionIndex-class
  (DiversityIndex), 2
CountMatrix, 5, 8, 23, 24

data.frame, 6, 21
dimensio::CA, 18
dimensio::ca(), 25
DiversityIndex, 2, 6, 10, 13, 18, 21
DiversityIndex-class (DiversityIndex), 2
eppm (independance), 8
eppm(), 12
eppm,CountMatrix-method (independance), 8
eppm-method (independance), 8
EvennessIndex-class (DiversityIndex), 2
Ford(plot_bar), 11
function, 11, 16
get (mutator), 9
get_index(mutator), 9
get_index, EvennessIndex-method (mutator), 9
get_index, HeterogeneityIndex-method (mutator), 9
get_index, RichnessIndex-method (mutator), 9
generate_index-method (mutator), 9
get_method(mutator), 9
get_method, DiversityIndex-method (mutator), 9
generate_method-method (mutator), 9
get_order(mutator), 9
generate_order, PermutationOrder-method (seriation), 22
generate_order-method (mutator), 9
ggplot2::ggplot, 11, 14, 15, 17
index_composition, CountMatrix-method (richness-index), 18
index_composition, IncidenceMatrix-method (richness-index), 18
index_composition-method (richness-index), 18
index_evenness (heterogeneity-index), 3
index_evenness, CountMatrix-method (heterogeneity-index), 3
index_evenness-method (heterogeneity-index), 3
index_heterogeneity (heterogeneity-index), 3
index_heterogeneity-method (heterogeneity-index), 3
index_richness (richness-index), 18
index_richness, CountMatrix-method (richness-index), 18
index_richness-method (richness-index), 18
integer, 3, 5, 10, 17, 18, 20, 24
jackknife_evenness (heterogeneity-index), 3
jackknife_evenness, CountMatrix-method (heterogeneity-index), 3
jackknife_evenness-method (heterogeneity-index), 3
jackknife_heterogeneity (heterogeneity-index), 3
jackknife_heterogeneity, CountMatrix-method (heterogeneity-index), 3
jackknife_richness (richness-index), 18
jackknife_richness, CountMatrix-method (richness-index), 18
jackknife_richness-method (richness-index), 18
list, 21, 31
logical, 5, 11, 14, 16, 20, 24, 30
matrigraphe (plot_matrix), 15
matrix, 8, 21, 29, 31
mutator, 9, 29
numeric, 3, 5, 8, 17, 18, 20, 24, 29
PermutationOrder, 23, 24
PermutationOrder-class, 10
permute (seriation), 22
permute, CountMatrix, PermutationOrder-method (seriation), 22
permute, IncidenceMatrix, PermutationOrder-method (seriation), 22
permute-method (seriation), 22
plot, DiversityIndex, missing-method (plot_diversity), 13
plot_bar, 11, 13–15, 17
plot_bertin (plot_bar), 11
plot_bertin, matrix-method (plot_bar), 11
plot_bertin-method (plot_bar), 11
plot_diversity, 12, 13, 14, 15, 17
plot_diversity(), 7, 22
plot_ford (plot_bar), 11
plot_ford(), 9
INDEX

plot_ford, CountMatrix-method (plot_bar), 11
plot_ford, matrix-method (plot_bar), 11
plot_ford-method (plot_bar), 11
plot_heatmap (plot_matrix), 15
plot_heatmap(), 9
plot_heatmap, matrix-method (plot_matrix), 15
plot_heatmap-method (plot_matrix), 15
plot_line, 12, 13, 14, 15, 17
plot_matrix, 12–14, 15, 17
plot_rank (plot_line), 14
plot_rank, matrix-method (plot_line), 14
plot_rank-method (plot_line), 14
plot_spot, 12–15, 16
plot_spot, dist-method (plot_spot), 16
plot_spot, matrix-method (plot_spot), 16
plot_spot, OccurrenceMatrix-method (plot_spot), 16
plot_spot-method (plot_spot), 16
pvi (independence), 8
pvi(), 15
pvi, CountMatrix-method (independence), 8
pvi-method (independence), 8

rarefaction (richness-index), 18
rarefaction, CountMatrix-method (richness-index), 18
rarefaction-method (richness-index), 18
refine_seriation (seriation), 22
refine_seriation, CA-method (seriation), 22
refine_seriation-method (seriation), 22
RefineCA, 24
RefineCA-class, 17
richness-index, 18
RichnessIndex-class (DiversityIndex), 2

seriate_average (seriation), 22
seriate_average, CountMatrix-method (seriation), 22
seriate_average, IncidenceMatrix-method (seriation), 22
seriate_average-method (seriation), 22
seriate_rank (seriation), 22
seriate_rank(), 9
seriate_rank, CountMatrix-method (seriation), 22

seriate_rank, IncidenceMatrix-method (seriation), 22
seriate_rank-method (seriation), 22
seriation, 22
seriographe (plot_bar), 11
set (mutator), 9
similarity, 7, 22, 26, 31
similarity(), 7
similarity, CountMatrix-method (similarity), 26
similarity, IncidenceMatrix-method (similarity), 26
similarity-method (similarity), 26
simulate_evenness (heterogeneity-index), 3
simulate_evenness, CountMatrix-method (heterogeneity-index), 3
simulate_evenness-method (heterogeneity-index), 3
simulate_heterogeneity (heterogeneity-index), 3
simulate_heterogeneity, CountMatrix-method (heterogeneity-index), 3
simulate_heterogeneity-method (heterogeneity-index), 3
simulate_richness (richness-index), 18
simulate_richness, CountMatrix-method (richness-index), 18
simulate_richness-method (richness-index), 18
stats::dist, 27
stats::p.adjust(), 29
stats::quantile(), 5, 20
subset, 10, 28

test_diversity, 9, 29
test_diversity, CountMatrix-method (test_diversity), 29
test_diversity-method (test_diversity), 29
turnover (turnover-index), 30
turnover(), 7
turnover, CountMatrix-method (turnover-index), 30
turnover, IncidenceMatrix-method (turnover-index), 30
turnover-index, 30
turnover-method (turnover-index), 30