Package: divent (via r-universe)

November 20, 2024

Type Package

Title Entropy Partitioning to Measure Diversity

Version 0.4-99.9013

Description Measurement and partitioning of diversity, based on Tsallis entropy, following Marcon and Herault (2015) <doi:10.18637/jss.v067.i08>. 'divent' provides functions to estimate alpha, beta and gamma diversity of communities, including phylogenetic and functional diversity.

URL `https://ericmarcon.github.io/divent/`, `https://github.com/EricMarcon/divent`

BugReports `https://github.com/EricMarcon/divent/issues`

License GNU General Public License

Depends R (>= 4.1), Rcpp (>= 0.12.14)

Imports alphahull, ape, cli, dbmss, dplyr, ggplot2, graphics, igraph, EntropyEstimation, RColorBrewer, RcppParallel, Rdpack, rlang, spatstat.explore, spatstat.geom, spatstat.random, stats, tibble, tidyr, vegan

Suggests ade4, knitr, pkgdown, rmarkdown, SPECIES, testthat (>= 3.0.0)

LinkingTo Rcpp, RcppParallel

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RdMacros Rdpack

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

SystemRequirements GNU make, pandoc

VignetteBuilder knitr

Config/pak/sysreqs libglpk-dev make libicu-dev libxml2-dev

Repository https://ericmarcon.r-universe.dev

RemoteUrl https://github.com/ericmarcon/divent

RemoteRef HEAD

RemoteSha 1db262ffe7cf94b1ca99fed9e54812d5823773ac

Contents

divent-package 3
abd_freq_count 4
abd_species 5
accum_div_phylo 6
accum_hill 9
accum_sp 12
accum_sp_hill 15
alphahull 18
autoplot.accumulation 19
autoplot.profile 20
autoplot.wmppp 21
coverage 22
div_hill 24
div_hurlbert 28
div_part 30
div_phylo 31
div_richness 34
div_similarity 36
ent_hurlbert 38
ent_phylo 40
ent_rao 42
ent_shannon 44
ent_similarity 47
ent_simpson 49
ent_sp_simpson 51
ent_tsallis 53
exp_q 55
e_n_q 56
fit_rac 57
fun_ordinariness 58
fun_similarity 60
ln_q 61
metacommunity 62
mock_3sp 63
paracou_6 64
paracou_6_fundist 65
paracou_6_taxo 65
phylo_divent 66
plot.phylo_divent 67
plot.species_distribution 67
probabilities 69

profile_hill 71
profile_phylo 74
profile_similarity 77
rcommunity 80
rlseries 84
species_distribution 85

Index **90**

`divent-package` *divent*

Description

Measures of Diversity and Entropy

Details

This package is a reboot of the **entropart** package (Marcon and Hérault 2015).

Author(s)

Maintainer: Eric Marcon `<eric.marcon@agroparistech.fr>` (ORCID)

References

Marcon E, Hérault B (2015). “Entropart, an R Package to Measure and Partition Diversity.” *Journal of Statistical Software*, **67**(8), 1–26. doi:10.18637/jss.v067.i08.

See Also

Useful links:

- `https://ericmarcon.github.io/divent/`
- `https://github.com/EricMarcon/divent`
- Report bugs at `https://github.com/EricMarcon/divent/issues`

abd_freq_count *Abundance Frequency Count of a Community*

Description

Count the number of species observed the same number of times.

Usage

```
abd_freq_count(
  abd,
  level = NULL,
  probability_estimator = c("naive", "Chao2013", "Chao2015", "ChaoShen"),
  unveiling = c("none", "uniform", "geometric"),
  richness_estimator = c("jackknife", "iChao1", "Chao1", "rarefy", "naive"),
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  check_arguments = TRUE
)
```

Arguments

`abd`	A numeric vector containing species abundances.
`level`	the level of interpolation or extrapolation. It may be a sample size (an integer) or a sample coverage (a number between 0 and 1). If not `NULL`, the asymptotic `estimator` is ignored.
`probability_estimator`	a string containing one of the possible estimators of the probability distribution (see probabilities). Used only for extrapolation.
`unveiling`	a string containing one of the possible unveiling methods to estimate the probabilities of the unobserved species (see probabilities). Used only for extrapolation.
`richness_estimator`	A string containing an estimator recognized by div_richness to evaluate the total number of species in probabilities. Used only for extrapolation.
`coverage_estimator`	an estimator of sample coverage used by coverage.
`check_arguments`	if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

The Abundance Frequency Count (Chao and Jost 2015) is the number of species observed each number of times. It is a way to summarize the species distribution.

It can be estimated at a specified level of interpolation or extrapolation. Extrapolation relies on the estimation of the estimation of the asymptotic distribution of the community by probabilities and eq. (5) of (Chao et al. 2014).

Value

A two-column tibble. The first column contains the number of observations, the second one the number of species observed this number of times.

References

Chao A, Gotelli NJ, Hsieh TC, Sander EL, Ma KH, Colwell RK, Ellison AM (2014). "Rarefaction and Extrapolation with Hill Numbers: A Framework for Sampling and Estimation in Species Diversity Studies." *Ecological Monographs*, **84**(1), 45–67. doi:10.1890/130133.1.

Chao A, Jost L (2015). "Estimating Diversity and Entropy Profiles via Discovery Rates of New Species." *Methods in Ecology and Evolution*, **6**(8), 873–882. doi:10.1111/2041210X.12349.

Examples

```
abd_freq_count(as.numeric(paracou_6_abd[1, ]))
```

---

abd_species *Abundances of Communities*

Description

Utilities for community abundances (objects of class "abundances").

Usage

```
abd_species(abundances, check_arguments = TRUE)

abd_sum(abundances, as_numeric = FALSE, check_arguments = TRUE)

prob_species(species_distribution, check_arguments = TRUE)
```

Arguments

`abundances`	an object of class abundances.
`check_arguments`	if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.
`as_numeric`	if `TRUE`, a number or a numeric vector is returned rather than a tibble.
`species_distribution`	an object of class species_distribution.

Value

`abd_species()` returns a tibble containing the species abundance columns only, to simplify numeric operations.

`prob_species()` returns the same tibble but values are probabilities.

`abd_sum()` returns the sample sizes of the communities in a numeric vector.

Examples

```
abd_species(paracou_6_abd)
abd_sum(paracou_6_abd)
```

`accum_div_phylo` *Phylogenetic Diversity Accumulation of a Community*

Description

Diversity and Entropy Accumulation Curves represent the accumulation of entropy with respect to the sample size.

Usage

```
accum_ent_phylo(x, ...)

## S3 method for class 'numeric'
accum_ent_phylo(
  x,
  tree,
  q = 0,
  normalize = TRUE,
  levels = NULL,
  probability_estimator = c("Chao2015", "Chao2013", "ChaoShen", "naive"),
  unveiling = c("geometric", "uniform", "none"),
  richness_estimator = c("rarefy", "jackknife", "iChao1", "Chao1", "naive"),
  jack_alpha = 0.05,
  jack_max = 10,
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  n_simulations = 0,
  alpha = 0.05,
  show_progress = TRUE,
  ...,
  check_arguments = TRUE
)

## S3 method for class 'abundances'
accum_ent_phylo(
  x,
  tree,
  q = 0,
  normalize = TRUE,
  levels = NULL,
  probability_estimator = c("Chao2015", "Chao2013", "ChaoShen", "naive"),
  unveiling = c("geometric", "uniform", "none"),
  richness_estimator = c("rarefy", "jackknife", "iChao1", "Chao1", "naive"),
  jack_alpha = 0.05,
  jack_max = 10,
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  gamma = FALSE,
  n_simulations = 0,
  alpha = 0.05,
  show_progress = TRUE,
  ...,
  check_arguments = TRUE
)

accum_div_phylo(x, ...)

## S3 method for class 'numeric'
accum_div_phylo(
  x,
  tree,
  q = 0,
  normalize = TRUE,
  levels = NULL,
  probability_estimator = c("Chao2015", "Chao2013", "ChaoShen", "naive"),
  unveiling = c("geometric", "uniform", "none"),
  richness_estimator = c("rarefy", "jackknife", "iChao1", "Chao1", "naive"),
  jack_alpha = 0.05,
  jack_max = 10,
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  n_simulations = 0,
  alpha = 0.05,
  show_progress = TRUE,
  ...,
  check_arguments = TRUE
)

## S3 method for class 'abundances'
accum_div_phylo(
  x,
  tree,
  q = 0,
  normalize = TRUE,
  levels = NULL,
  probability_estimator = c("Chao2015", "Chao2013", "ChaoShen", "naive"),
  unveiling = c("geometric", "uniform", "none"),
  richness_estimator = c("rarefy", "jackknife", "iChao1", "Chao1", "naive"),
  jack_alpha = 0.05,
  jack_max = 10,
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  gamma = FALSE,
  n_simulations = 0,
  alpha = 0.05,
  show_progress = TRUE,
  ...,
  check_arguments = TRUE
)
```

Arguments

`x`	An object, that may be a numeric vector containing abundances or probabilities, or an object of class abundances or probabilities.
`...`	Unused.
`tree`	an ultrametric, phylogenetic tree. May be an object of class phylo_divent, ape::phylo, ade4::phylog or stats::hclust.
`q`	a number: the order of diversity.
`normalize`	if `TRUE`, phylogenetic is normalized: the height of the tree is set to 1.
`levels`	The levels, i.e. the sample sizes of interpolation or extrapolation: a vector of integer values.
`probability_estimator`	a string containing one of the possible estimators of the probability distribution (see probabilities). Used only for extrapolation.
`unveiling`	a string containing one of the possible unveiling methods to estimate the probabilities of the unobserved species (see probabilities). Used only for extrapolation.
`richness_estimator`	an estimator of richness to evaluate the total number of species, see div_richness. used for interpolation and extrapolation.
`jack_alpha`	the risk level, 5% by default, used to optimize the jackknife order.
`jack_max`	the highest jackknife order allowed. Default is 10.
`coverage_estimator`	an estimator of sample coverage used by coverage.
`n_simulations`	the number of simulations used to estimate the confidence envelope.
`alpha`	the risk level, 5% by default.
`show_progress`	if TRUE, a progress bar is shown during long computations.
`check_arguments`	if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.
`gamma`	if `TRUE`, γ diversity, i.e. diversity of the metacommunity, is computed.

Details

`accum_ent_phylo()` or `accum_div_phylo()` estimate the phylogenetic diversity or entropy accumulation curve of a distribution. See ent_tsallis for details about the computation of entropy at each level of interpolation and extrapolation.

In accumulation curves, extrapolation if done by estimating the asymptotic distribution of the community and estimating entropy at different levels by interpolation.

Interpolation and extrapolation of integer orders of diversity are from Chao et al. (2014). The asymptotic richness is adjusted so that the extrapolated part of the accumulation joins the observed value at the sample size.

"accumulation" objects can be plotted. They generalize the classical Species Accumulation Curves (SAC) which are diversity accumulation of order $q = 0$.

Value

A tibble with the site names, the estimators used and the accumulated entropy or diversity at each level of sampling effort.

References

Chao A, Gotelli NJ, Hsieh TC, Sander EL, Ma KH, Colwell RK, Ellison AM (2014). “Rarefaction and Extrapolation with Hill Numbers: A Framework for Sampling and Estimation in Species Diversity Studies.” *Ecological Monographs*, **84**(1), 45–67. doi:10.1890/130133.1.

Examples

```
# Richness accumulation up to the sample size.
# 100 simulations only to save time.
autoplot(
  accum_div_phylo(mock_3sp_abd, tree = mock_3sp_tree, n_simulations = 100)
)
```

`accum_hill` *Diversity Accumulation of a Community*

Description

Diversity and Entropy Accumulation Curves represent the accumulation of entropy and diversity with respect to the sample size.

Usage

```
accum_tsallis(x, ...)

## S3 method for class 'numeric'
accum_tsallis(
  x,
  q = 0,
  levels = seq_len(sum(x)),
  probability_estimator = c("Chao2015", "Chao2013", "ChaoShen", "naive"),
  unveiling = c("geometric", "uniform", "none"),
  richness_estimator = c("rarefy", "jackknife", "iChao1", "Chao1", "naive"),
  jack_alpha = 0.05,
  jack_max = 10,
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  n_simulations = 0,
  alpha = 0.05,
  show_progress = TRUE,
  ...,
  check_arguments = TRUE
)

## S3 method for class 'abundances'
accum_tsallis(
  x,
  q = 0,
  levels = NULL,
  probability_estimator = c("Chao2015", "Chao2013", "ChaoShen", "naive"),
  unveiling = c("geometric", "uniform", "none"),
  richness_estimator = c("rarefy", "jackknife", "iChao1", "Chao1", "naive"),
  jack_alpha = 0.05,
  jack_max = 10,
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  n_simulations = 0,
  alpha = 0.05,
  show_progress = TRUE,
  ...,
  check_arguments = TRUE
)

accum_hill(x, ...)

## S3 method for class 'numeric'
accum_hill(
  x,
  q = 0,
  levels = seq_len(sum(x)),
  probability_estimator = c("Chao2015", "Chao2013", "ChaoShen", "naive"),
  unveiling = c("geometric", "uniform", "none"),
  richness_estimator = c("rarefy", "jackknife", "iChao1", "Chao1", "naive"),
  jack_alpha = 0.05,
  jack_max = 10,
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  n_simulations = 0,
  alpha = 0.05,
  show_progress = TRUE,
  ...,
  check_arguments = TRUE
)

## S3 method for class 'abundances'
accum_hill(
  x,
  q = 0,
  levels = NULL,
  probability_estimator = c("Chao2015", "Chao2013", "ChaoShen", "naive"),
  unveiling = c("geometric", "uniform", "none"),
  richness_estimator = c("rarefy", "jackknife", "iChao1", "Chao1", "naive"),
  jack_alpha = 0.05,
  jack_max = 10,
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  n_simulations = 0,
  alpha = 0.05,
  show_progress = TRUE,
  ...,
  check_arguments = TRUE
)
```

Arguments

`x`	An object, that may be a numeric vector containing abundances or probabilities, or an object of class abundances or probabilities.
`...`	Unused.
`q`	a number: the order of diversity.
`levels`	The levels, i.e. the sample sizes of interpolation or extrapolation: a vector of integer values.
`probability_estimator`	a string containing one of the possible estimators of the probability distribution (see probabilities). Used only for extrapolation.
`unveiling`	a string containing one of the possible unveiling methods to estimate the probabilities of the unobserved species (see probabilities). Used only for extrapolation.
`richness_estimator`	an estimator of richness to evaluate the total number of species, see div_richness. used for interpolation and extrapolation.
`jack_alpha`	the risk level, 5% by default, used to optimize the jackknife order.
`jack_max`	the highest jackknife order allowed. Default is 10.
`coverage_estimator`	an estimator of sample coverage used by coverage.
`n_simulations`	the number of simulations used to estimate the confidence envelope.
`alpha`	the risk level, 5% by default.
`show_progress`	if TRUE, a progress bar is shown during long computations.
`check_arguments`	if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

`accum_hill()` or `accum_tsallis()` estimate the diversity or entropy accumulation curve of a distribution. See ent_tsallis for details about the computation of entropy at each level of interpolation and extrapolation.

In accumulation curves, extrapolation is done by estimating the asymptotic distribution of the community and estimating entropy at different levels by interpolation.

Interpolation and extrapolation of integer orders of diversity are from Chao et al. (2014). The asymptotic richness is adjusted so that the extrapolated part of the accumulation joins the observed value at the sample size.

"accumulation" objects can be plotted. They generalize the classical Species Accumulation Curves (SAC) which are diversity accumulation of order $q = 0$.

Value

A tibble with the site names, the estimators used and the accumulated entropy or diversity at each level of sampling effort.

References

Chao A, Gotelli NJ, Hsieh TC, Sander EL, Ma KH, Colwell RK, Ellison AM (2014). "Rarefaction and Extrapolation with Hill Numbers: A Framework for Sampling and Estimation in Species Diversity Studies." *Ecological Monographs*, **84**(1), 45–67. doi:10.1890/130133.1.

Examples

```
# Paracou 6 subplot 1
autoplot(accum_hill(paracou_6_abd[1, ]))
```

accum_sp *Spatial Accumulation of Diversity*

Description

A spatial accumulation is a measure of diversity with respect to the distance from individuals.

Usage

```
## S3 method for class 'accum_sp'
plot(
  x,
  ...,
  q = dimnames(x$accumulation)$q[1],
  type = "l",
  main = "accumulation of ...",
  xlab = "Sample size...",
  ylab = "Diversity...",
  ylim = NULL,
  show_h0 = TRUE,
  line_width = 2,
  col_shade = "grey75",
  col_border = "red"
)

## S3 method for class 'accum_sp'
autoplot(
  object,
  ...,
  q = dimnames(object$accumulation)$q[1],
  main = "Accumulation of ...",
  xlab = "Sample size...",
  ylab = "Diversity...",
  ylim = NULL,
  show_h0 = TRUE,
  col_shade = "grey75",
  col_border = "red"
)

plot_map(
  accum,
  q = dimnames(accum$accumulation)$q[1],
  neighborhood = dplyr::last(colnames(accum$neighborhoods)),
  sigma = spatstat.explore::bw.scott(accum$X, isotropic = TRUE),
  allow_jitter = TRUE,
  weighted = FALSE,
  adjust = 1,
  dim_x = 128,
  dim_y = 128,
  main = "",
  col = grDevices::terrain.colors(256),
  contour = TRUE,
  contour_levels = 10,
  contour_col = "dark red",
  points = FALSE,
  pch = 20,
  point_col = "black",
  suppress_margins = TRUE,
  ...,
  check_arguments = TRUE
)
```

Arguments

x	an accum_sp object.

`...`	Additional arguments to be passed to plot, or, in `plot_map()`, to spatstat.explore::bw.smoothppp and spatstat.explore::density.ppp to control the kernel smoothing and to spatstat.geom::plot.im to plot the image.
`q`	a number: the order of diversity.
`type`	plotting parameter. Default is "l".
`main`	main title of the plot.
`xlab`	X-axis label.
`ylab`	Y-axis label.
`ylim`	limits of the Y-axis, as a vector of two numeric values.
`show_h0`	if `TRUE`, the values of the null hypothesis are plotted.
`line_width`	width of the Diversity Accumulation Curve line.
`col_shade`	The color of the shaded confidence envelope.
`col_border`	The color of the borders of the confidence envelope.
`object`	an `accum_sp` object.
`accum`	an object to map.
`neighborhood`	The neighborhood size, i.e. the number of neighbors or the distance to consider.
`sigma`	the smoothing bandwidth. The standard deviation of the isotropic smoothing kernel. Either a numerical value, or a function that computes an appropriate value of sigma.
`allow_jitter`	if `TRUE`, duplicated points are jittered to avoid their elimination by the smoothing procedure.
`weighted`	if `TRUE`, the weight of the points is used by the smoothing procedure.
`adjust`	force the automatically selected bandwidth to be multiplied by `adjust`. Setting it to values lower than one (1/2 for example) will sharpen the estimation.
`dim_x`	the number of columns (pixels) of the resulting map, 128 by default.
`dim_y`	the number of rows (pixels) of the resulting map, 128 by default.
`col`	the colors of the map. See spatstat.geom::plot.im for details.
`contour`	if `TRUE`, contours are added to the map.
`contour_levels`	the number of levels of contours.
`contour_col`	the color of the contour lines.
`points`	if `TRUE`, the points that brought the data are added to the map.
`pch`	the symbol used to represent points.
`point_col`	the color of the points. Standard base graphic arguments such as `main` can be used.
`suppress_margins`	if `TRUE`, the map has reduced margins.
`check_arguments`	if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

Objects of class `accum_sp` contain the value of diversity (`accum_sp_diversity` objects), entropy (`accum_sp_entropy` objects) or mixing (`accum_sp_mixing` objects) at distances from the individuals.

These objects are lists:

- `X` contains the dbmss::wmppp point pattern,
- `accumulation` is a 3-dimensional array, with orders of diveristy in rows, neighborhood size (number of points or distance) in columns and a single slice for the observed entropy, diversity or mixing.
- `neighborhoods` is a similar 3-dimensional array with one slice per point of `X`.

They can be plotted or mapped.

Value

`plot.accum_sp()` returns `NULL`.

`autoplot.accum_sp()` returns a ggplot2::ggplot object.

`plot_map` returns a spatstat.geom::im object that can be used to produce alternative maps.

Examples

```
# Generate a random community
X <- rspcommunity(1, size = 50, species_number = 10)
# Calculate the species accumulation curve
accum <- accum_sp_hill(X, orders = 0, r = c(0, 0.2), individual = TRUE)
# Plot the local richness at distance = 0.2
plot_map(accum, q = 0, neighborhood = 0.2)
```

`accum_sp_hill` *Spatial Diversity Accumulation of a Community*

Description

Spatial Diversity and Entropy Accumulation Curves represent the accumulation of entropy and diversity with respect to the distance from individuals

Usage

```
accum_sp_tsallis(
  X,
  orders = 0,
  neighbors = 1:ceiling(X$n/2),
  r = NULL,
  correction = c("none", "extrapolation"),
  richness_estimator = c("rarefy", "jackknife", "iChao1", "Chao1", "naive"),
  individual = FALSE,
  show_progress = TRUE,
  check_arguments = TRUE
)

accum_sp_hill(
  X,
  orders = 0,
  neighbors = 1:ceiling(X$n/2),
  r = NULL,
  correction = c("none", "extrapolation"),
  richness_estimator = c("rarefy", "jackknife", "iChao1", "Chao1", "naive"),
  h0 = c("none", "multinomial", "random location", "binomial"),
  alpha = 0.05,
  n_simulations = 100,
  individual = FALSE,
  show_progress = TRUE,
  check_arguments = TRUE
)

accum_mixing(
  X,
  orders = 0,
  neighbors = 1:ceiling(X$n/2),
  r = NULL,
  correction = c("none", "extrapolation"),
  richness_estimator = c("rarefy", "jackknife", "iChao1", "Chao1", "naive"),
  h0 = c("none", "multinomial", "random location", "binomial"),
  alpha = 0.05,
  n_simulations = 100,
  individual = FALSE,
  show_progress = TRUE,
  check_arguments = TRUE
)
```

Arguments

`X`	a spatialized community (A dbmss::wmppp object with `PointType` values as species names.)
`orders`	A numeric vector: the diversity orders to address. Default is 0.
`neighbors`	A vector of integers. Entropy will be accumulated along this number of neighbors around each individual. Default is 10% of the individuals.
`r`	A vector of distances. If `NULL` accumulation is along `n`, else neighbors are accumulated in circles of radius `r`.
`correction`	The edge-effect correction to apply when estimating the entropy of a neighborhood community that does not fit in the window. Does not apply if neighborhoods are defined by the number of neighbors. Default is "none". "extrapola-

tion" extrapolates the observed diversity up to the number of individuals estimated in the full area of the neighborhood, which is slow.

`richness_estimator`	an estimator of richness to evaluate the total number of species, see div_richness. used for interpolation and extrapolation.
`individual`	If `TRUE`, individual neighborhood entropies are returned.
`show_progress`	if TRUE, a progress bar is shown during long computations.
`check_arguments`	if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.
`h0`	The null hypothesis to compare the distribution of `X` to. If "none", the default value, no null hypothesis is tested. "multinomial" means the community will be rarefied down to the number of `neighbors`. "random location" means the points will we randomly permuted across their actual locations. "binomial" means the points will we uniformly and independently drawn in the window (a binomial point process is a Poisson point process conditionally to the number of points).
`alpha`	the risk level, 5% by default.
`n_simulations`	the number of simulations used to estimate the confidence envelope.

Details

`accum_sp_hill()` or `accum_sp_tsallis()` estimate the diversity or entropy accumulation curve of a distribution.

Value

An accum_sp object, that is also either an accum_sp_diversity, accum_sp_entropy or accum_sp_mixing object.

Examples

```
# Generate a random community
X <- rspcommunity(1, size = 50, species_number = 3)
# Calculate the accumulation of richness
accum <- accum_sp_hill(X)
plot(accum, q = 0)
# along distance
accum_r <- accum_sp_hill(X, orders = 1, r = seq(0, .5, .05))
autoplot(accum_r, q = 1)
```

alphahull *alpha-shape calculation*

Description

Calculate a window containing all points of a point pattern. The window is not convex but as close as possible to the points.

Usage

```
alphahull(X, alpha = NULL)
```

Arguments

`X`	A planar point pattern (spatstat.geom::ppp.object).
`alpha`	A smoothing parameter to delimit concave polygons.

Details

The typical use of this function is to define a narrow window around a point pattern that has been created with a default, rectangle window.

The window is built by the `alphahull::ashape()` function and then transformed into a spatstat.geom::owin.object. The `alpha` parameter determines the smallest size of zones excluded from the window. If it is not specified, a first attempt is 1/256 of the diameter of the existing window of `X`. If the shape cannot be calculated, `alpha` is doubled and a new attempt is made.

Value

A window, i.e. a spatstat.geom::owin.object.

See Also

spatstat.geom::convexhull

Examples

```
# Simulate a point pattern
if (require(spatstat.random)) {
  X <- rpoispp(50)
  plot(X)
  # Calculate its border
  X$window <- alphahull(X)
  plot(X)
}
```

`autoplot.accumulation` *Plot Accumulation Objects*

Description

Plot objects of class "accumulation" produced by accum_hill and other accumulation functions.

Usage

```
## S3 method for class 'accumulation'
autoplot(
  object,
  ...,
  main = NULL,
  xlab = "Sample Size",
  ylab = NULL,
  shade_color = "grey75",
  alpha = 0.3,
  lty = ggplot2::GeomLine$default_aes$linetype,
  lwd = ggplot2::GeomLine$default_aes$linewidth
)
```

Arguments

`object`	An object of class "accumulation".
`...`	Unused.
`main`	The main title of the plot.
`xlab`	The label of the x-axis.
`ylab`	The label of the y-axis.
`shade_color`	The color of the shaded confidence envelopes.
`alpha`	The opacity of the confidence envelopes, between 0 (transparent) and 1 (opaque).
`lty`	The line type of the curves.
`lwd`	The line width of the curves.

Value

A ggplot2::ggplot object.

Examples

```
# Species accumulation curve
autoplot(accum_hill(mock_3sp_abd))
```

`autoplot.profile` *Plot Profile Objects*

Description

Plot objects of class "profile" produced by profile_hill and other profile functions.

Usage

```
## S3 method for class 'profile'
autoplot(
  object,
  ...,
  main = NULL,
  xlab = "Order of Diversity",
  ylab = "Diversity",
  shade_color = "grey75",
  alpha = 0.3,
  lty = ggplot2::GeomLine$default_aes$linetype,
  lwd = ggplot2::GeomLine$default_aes$linewidth
)
```

Arguments

`object`	An object of class "profile".
`...`	Unused.
`main`	The main title of the plot.
`xlab`	The label of the x-axis.
`ylab`	The label of the y-axis.
`shade_color`	The color of the shaded confidence envelopes.
`alpha`	The opacity of the confidence envelopes, between 0 (transparent) and 1 (opaque).
`lty`	The line type of the curves.
`lwd`	The line width of the curves.

Value

A ggplot2::ggplot object.

Examples

```
# Diversity profile curve
autoplot(profile_hill(mock_3sp_abd))
```

`autoplot.wmppp` *ggplot method to plot wmppp objects*

Description

This method is from the dbmss package. See dbmss::autoplot.wmppp.

Usage

```
## S3 method for class 'wmppp'
autoplot(
  object,
  ...,
  show.window = TRUE,
  MaxPointTypes = 6,
  Other = "Other",
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  LegendLabels = NULL,
  labelSize = "Weight",
  labelColor = "Type",
  palette = "Set1",
  windowColor = "black",
  windowFill = "transparent",
  alpha = 1
)
```

Arguments

`object`	an object to be plotted.
`...`	extra arguments, currently unused.
`show.window`	if `TRUE`, the borders of the window containing the points are shown on the point map.
`MaxPointTypes`	the maximum number of different point types to show. If the point set contains more of them, the less frequent ones are gathered as "Other". This number must be limited for readability and not to exceed the number of colors offered by the palette.
`Other`	the name of the point types gathered as "Other"
`main`	the title of the plot.
`xlab`	the X-axis label.
`ylab`	the Y-axis label.
`LegendLabels`	a vector of characters. The first two items describe the observed and null-hypothesis curves, the third and last item the confidence interval. To be used only in plots with two curves (typically observed and expected values). The default is `NULL` to display the full description of functions.

`labelSize`	the guide of the point size legend in point maps, i.e. what the `PointSize` mark represents.
`labelColor`	the guide of the point color legend in point maps, i.e. what the `PointType` mark represents.
`palette`	The color palette used to display point types in maps. See ggplot2::scale_colour_brewer.
`windowColor`	the color used to draw the limits of the windows in point maps.
`windowFill`	the color used to fill the windows in point maps.
`alpha`	the opacity of the confidence envelope (in function values) or the points (in maps), between 0 and 1.

Value

A ggplot2::ggplot.

Examples

```
autoplot(paracou_6_wmppp)
```

`coverage` *Sample Coverage of a Community*

Description

`coverage()` calculates an estimator of the sample coverage of a community described by its abundance vector. `coverage_to_size()` estimates the sample size corresponding to the chosen sample coverage.

Usage

```
coverage(x, ...)

## S3 method for class 'numeric'
coverage(
  x,
  estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  level = NULL,
  as_numeric = FALSE,
  ...,
  check_arguments = TRUE
)

## S3 method for class 'abundances'
coverage(
  x,
  estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  level = NULL,
  ...,
  check_arguments = TRUE
)

coverage_to_size(x, ...)

## S3 method for class 'numeric'
coverage_to_size(
  x,
  sample_coverage,
  estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  as_numeric = FALSE,
  ...,
  check_arguments = TRUE
)

## S3 method for class 'abundances'
coverage_to_size(
  x,
  sample_coverage,
  estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  ...,
  check_arguments = TRUE
)
```

Arguments

`x`	An object.
`...`	Unused.
`estimator`	An estimator of the sample coverage. "ZhangHuang" is the most accurate but does not allow choosing a `level`. "Good"'s estimator only allows interpolation, i.e. estimation of the coverage of a subsample. "Chao" allows estimation at any `level`, including extrapolation. "Turing" is the simplest estimator, equal to 1 minus the number of singletons divided by the sample size.
`level`	The level of interpolation or extrapolation, i.e. an abundance.
`as_numeric`	if `TRUE`, a number or a numeric vector is returned rather than a tibble.
`check_arguments`	if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.
`sample_coverage`	The target sample coverage.

Details

The sample coverage C of a community is the total probability of occurrence of the species observed in the sample. $1 - C$ is the probability for an individual of the whole community to belong to a species that has not been sampled.

The historical estimator is due to Turing (Good 1953). It only relies on singletons (species observed only once). Chao's (Chao and Shen 2010) estimator uses doubletons too and Zhang-Huang's (Chao et al. 1988; Zhang and Huang 2007) uses the whole distribution.

If `level` is not `NULL`, the sample coverage is interpolated or extrapolated. Interpolation by the Good estimator relies on the equality between sampling deficit and the generalized Simpson entropy (Good 1953). The Chao et al. (2014) estimator allows extrapolation, reliable up a level equal to the double size of the sample.

Value

`coverage()` returns a named number equal to the calculated sample coverage. The name is that of the estimator used.

`coverage_to_size()` returns a number equal to the sample size corresponding to the chosen sample coverage.

References

Chao A, Gotelli NJ, Hsieh TC, Sander EL, Ma KH, Colwell RK, Ellison AM (2014). "Rarefaction and Extrapolation with Hill Numbers: A Framework for Sampling and Estimation in Species Diversity Studies." *Ecological Monographs*, **84**(1), 45–67. doi:10.1890/130133.1.

Chao A, Lee S, Chen T (1988). "A Generalized Good's Nonparametric Coverage Estimator." *Chinese Journal of Mathematics*, **16**, 189–199. 43836340.

Chao A, Shen T (2010). *Program SPADE: Species Prediction and Diversity Estimation. Program and User's Guide.*. CARE.

Good IJ (1953). "The Population Frequency of Species and the Estimation of Population Parameters." *Biometrika*, **40**(3/4), 237–264. doi:10.1093/biomet/40.34.237.

Zhang Z, Huang H (2007). "Turing's Formula Revisited." *Journal of Quantitative Linguistics*, **14**(2-3), 222–241. doi:10.1080/09296170701514189.

Examples

```
coverage(paracou_6_abd)
coverage_to_size(paracou_6_abd, sample_coverage = 0.9)
```

`div_hill` *Hill number of a Community*

Description

Estimate the diversity sensu stricto, i.e. the Hill (1973) number of species from abundance or probability data.

Usage

```
div_hill(x, q = 1, ...)

## S3 method for class 'numeric'
div_hill(
  x,
  q = 1,
 estimator = c("UnveilJ", "ChaoJost", "ChaoShen", "GenCov", "Grassberger", "Marcon",
    "UnveilC", "UnveiliC", "ZhangGrabchak", "naive", "Bonachela", "Holste"),
  level = NULL,
  probability_estimator = c("Chao2015", "Chao2013", "ChaoShen", "naive"),
  unveiling = c("geometric", "uniform", "none"),
  richness_estimator = c("jackknife", "iChao1", "Chao1", "naive"),
  jack_alpha = 0.05,
  jack_max = 10,
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  sample_coverage = NULL,
  as_numeric = FALSE,
  ...,
  check_arguments = TRUE
)

## S3 method for class 'species_distribution'
div_hill(
  x,
  q = 1,
 estimator = c("UnveilJ", "ChaoJost", "ChaoShen", "GenCov", "Grassberger", "Marcon",
    "UnveilC", "UnveiliC", "ZhangGrabchak", "naive", "Bonachela", "Holste"),
  level = NULL,
  probability_estimator = c("Chao2015", "Chao2013", "ChaoShen", "naive"),
  unveiling = c("geometric", "uniform", "none"),
  richness_estimator = c("jackknife", "iChao1", "Chao1", "naive"),
  jack_alpha = 0.05,
  jack_max = 10,
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  gamma = FALSE,
  as_numeric = FALSE,
  ...,
  check_arguments = TRUE
)
```

Arguments

x	An object, that may be a numeric vector containing abundances or probabilities, or an object of class abundances or probabilities.
q	a number: the order of diversity.
...	Unused.

`estimator`	An estimator of asymptotic diversity.
`level`	the level of interpolation or extrapolation. It may be a sample size (an integer) or a sample coverage (a number between 0 and 1). If not `NULL`, the asymptotic `estimator` is ignored.
`probability_estimator`	a string containing one of the possible estimators of the probability distribution (see probabilities). Used only for extrapolation.
`unveiling`	a string containing one of the possible unveiling methods to estimate the probabilities of the unobserved species (see probabilities). Used only for extrapolation.
`richness_estimator`	an estimator of richness to evaluate the total number of species, see div_richness. used for interpolation and extrapolation.
`jack_alpha`	the risk level, 5% by default, used to optimize the jackknife order.
`jack_max`	the highest jackknife order allowed. Default is 10.
`coverage_estimator`	an estimator of sample coverage used by coverage.
`sample_coverage`	the sample coverage of `x` calculated elsewhere. Used to calculate the gamma diversity of meta-communities, see details.
`as_numeric`	if `TRUE`, a number or a numeric vector is returned rather than a tibble.
`check_arguments`	if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.
`gamma`	if `TRUE`, γ diversity, i.e. diversity of the metacommunity, is computed.

Details

Several estimators are available to deal with incomplete sampling.

Bias correction requires the number of individuals.

Estimation techniques are from Chao and Shen (2003), Grassberger (1988),Holste et al. (1998), Bonachela et al. (2008), Marcon et al. (2014) which is actually the max value of "ChaoShen" and "Grassberger", Zhang and Grabchak (2014), Chao et al. (2015), Chao and Jost (2015) and Marcon (2015).

The `ChaoJost` estimator Chao et al. (2013); Chao and Jost (2015) contains an unbiased part concerning observed species, equal to that of Zhang and Grabchak (2014), and a (biased) estimator of the remaining bias based on the estimation of the species-accumulation curve. It is very efficient but slow if the number of individuals is more than a few hundreds.

The unveiled estimators rely on Chao et al. (2015), completed by Marcon (2015). The actual probabilities of observed species are estimated and completed by a geometric distribution of the probabilities of unobserved species. The number of unobserved species is estimated by the Chao1 estimator (`UnveilC`), following Chao et al. (2015), or by the iChao1 (`UnveiliC`) or the jackknife (`UnveilJ`). The `UnveilJ` estimator often has a lower bias but a greater variance (Marcon 2015). It is a good first choice thanks to the versatility of the jackknife estimator of richness.

Estimators by Bonachela et al. (2008) and Holste et al. (1998) are rarely used.

To estimate γ diversity, the size of a metacommunity (see metacommunity) is unknown so it has to be set according to a rule which does not ensure that its abundances are integer values. Then, classical bias-correction methods do not apply. Providing the `sample_coverage` argument allows applying the `ChaoShen` and `Grassberger` estimators to estimate quite well the entropy.

Diversity can be estimated at a specified level of interpolation or extrapolation, either a chosen sample size or sample coverage (Chao et al. 2014), rather than its asymptotic value. See accum_hill for details.

Value

A tibble with the site names, the estimators used and the estimated diversity.

References

Bonachela JA, Hinrichsen H, Muñoz MA (2008). "Entropy Estimates of Small Data Sets." *Journal of Physics A: Mathematical and Theoretical*, **41**(202001), 1–9. doi:10.1088/17518113/41/20/202001.

Chao A, Gotelli NJ, Hsieh TC, Sander EL, Ma KH, Colwell RK, Ellison AM (2014). "Rarefaction and Extrapolation with Hill Numbers: A Framework for Sampling and Estimation in Species Diversity Studies." *Ecological Monographs*, **84**(1), 45–67. doi:10.1890/130133.1.

Chao A, Hsieh TC, Chazdon RL, Colwell RK, Gotelli NJ (2015). "Unveiling the Species-Rank Abundance Distribution by Generalizing Good-Turing Sample Coverage Theory." *Ecology*, **96**(5), 1189–1201. doi:10.1890/140550.1.

Chao A, Jost L (2015). "Estimating Diversity and Entropy Profiles via Discovery Rates of New Species." *Methods in Ecology and Evolution*, **6**(8), 873–882. doi:10.1111/2041210X.12349.

Chao A, Shen T (2003). "Nonparametric Estimation of Shannon's Index of Diversity When There Are Unseen Species in Sample." *Environmental and Ecological Statistics*, **10**(4), 429–443. doi:10.1023/A:1026096204727.

Chao A, Wang Y, Jost L (2013). "Entropy and the Species Accumulation Curve: A Novel Entropy Estimator via Discovery Rates of New Species." *Methods in Ecology and Evolution*, **4**(11), 1091–1100. doi:10.1111/2041210x.12108.

Grassberger P (1988). "Finite Sample Corrections to Entropy and Dimension Estimates." *Physics Letters A*, **128**(6-7), 369–373. doi:10.1016/03759601(88)901934.

Hill MO (1973). "Diversity and Evenness: A Unifying Notation and Its Consequences." *Ecology*, **54**(2), 427–432. doi:10.2307/1934352.

Holste D, Große I, Herzel H (1998). "Bayes' Estimators of Generalized Entropies." *Journal of Physics A: Mathematical and General*, **31**(11), 2551–2566.

Marcon E (2015). "Practical Estimation of Diversity from Abundance Data." *HAL*, **01212435**(version 2).

Marcon E, Scotti I, Hérault B, Rossi V, Lang G (2014). "Generalization of the Partitioning of Shannon Diversity." *Plos One*, **9**(3), e90289. doi:10.1371/journal.pone.0090289.

Zhang Z, Grabchak M (2014). "Nonparametric Estimation of Kullback-Leibler Divergence." *Neural computation*, **26**(11), 2570–2593. doi:10.1162/NECO_a_00646, 25058703.

Examples

```
# Diversity of each community
div_hill(paracou_6_abd, q = 2)
# gamma diversity
div_hill(paracou_6_abd, q = 2, gamma = TRUE)

# At 80% coverage
div_hill(paracou_6_abd, q = 2, level = 0.8)
```

---

div_hurlbert *Hurlbert Diversity of a Community*

Description

Estimate the diversity sensu stricto, i.e. the effective number of species number of species Dauby and Hardy (2012) from abundance or probability data.

Usage

```
div_hurlbert(x, k = 1, ...)

## S3 method for class 'numeric'
div_hurlbert(
  x,
  k = 2,
  estimator = c("Hurlbert", "naive"),
  as_numeric = FALSE,
  ...,
  check_arguments = TRUE
)

## S3 method for class 'species_distribution'
div_hurlbert(
  x,
  k = 2,
  estimator = c("Hurlbert", "naive"),
  as_numeric = FALSE,
  ...,
  check_arguments = TRUE
)
```

Arguments

`x`	An object, that may be a numeric vector containing abundances or probabilities, or an object of class abundances or probabilities.
`k`	the order of Hurlbert's diversity.
`...`	Unused.
`estimator`	An estimator of asymptotic diversity.
`as_numeric`	if `TRUE`, a number or a numeric vector is returned rather than a tibble.
`check_arguments`	if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

Several estimators are available to deal with incomplete sampling.

Bias correction requires the number of individuals.

Estimation techniques are from Hurlbert (1971).

Hurlbert's diversity cannot be estimated at a specified level of interpolation or extrapolation, and diversity partioning is not available.

Value

A tibble with the site names, the estimators used and the estimated diversity.

References

Dauby G, Hardy OJ (2012). "Sampled-Based Estimation of Diversity Sensu Stricto by Transforming Hurlbert Diversities into Effective Number of Species." *Ecography*, **35**(7), 661–672. doi:10.1111/j.16000587.2011.06860.x.

Hurlbert SH (1971). "The Nonconcept of Species Diversity: A Critique and Alternative Parameters." *Ecology*, **52**(4), 577–586. doi:10.2307/1934145.

Examples

```
# Diversity of each community
div_hurlbert(paracou_6_abd, k = 2)
```

div_part *Diversity partition*

Description

Calculate γ, β and α diversities of a metacommunity.

Usage

```
div_part(
  abundances,
  q = 1,
 estimator = c("UnveilJ", "ChaoJost", "ChaoShen", "GenCov", "Grassberger", "Holste",
    "Marcon", "UnveilC", "UnveiliC", "ZhangGrabchak"),
  level = NULL,
  probability_estimator = c("Chao2015", "Chao2013", "ChaoShen", "naive"),
  unveiling = c("geometric", "uniform", "none"),
  richness_estimator = c("jackknife", "iChao1", "Chao1", "naive"),
  jack_alpha = 0.05,
  jack_max = 10,
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  check_arguments = TRUE
)
```

Arguments

`abundances`	an object of class abundances.
`q`	a number: the order of diversity.
`estimator`	An estimator of diversity.
`level`	the level of interpolation or extrapolation. It may be a sample size (an integer) or a sample coverage (a number between 0 and 1). If not `NULL`, the asymptotic `estimator` is ignored.
`probability_estimator`	a string containing one of the possible estimators of the probability distribution (see probabilities). Used only for extrapolation.
`unveiling`	a string containing one of the possible unveiling methods to estimate the probabilities of the unobserved species (see probabilities). Used only for extrapolation.
`richness_estimator`	an estimator of richness to evaluate the total number of species, see div_richness. used for interpolation and extrapolation.
`jack_alpha`	the risk level, 5% by default, used to optimize the jackknife order.
`jack_max`	the highest jackknife order allowed. Default is 10.
`coverage_estimator`	an estimator of sample coverage used by coverage.

`check_arguments`
: if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

The function computes γ diversity after building a metacommunity from local communities according to their weight (Marcon et al. 2014). α entropy is the weighted mean local entropy, converted into Hill numbers to obtain α diversity. β diversity is obtained as the ratio of γ to α.

Value

A tibble with diversity values at each scale.

References

Marcon E, Scotti I, Hérault B, Rossi V, Lang G (2014). "Generalization of the Partitioning of Shannon Diversity." *Plos One*, **9**(3), e90289. doi:10.1371/journal.pone.0090289.

Examples

```
div_part(paracou_6_abd)
```

`div_phylo` *Phylogenetic Diversity of a Community*

Description

Estimate the diversity of species from abundance or probability data and a phylogenetic tree. Several estimators are available to deal with incomplete sampling.

Usage

```
div_phylo(x, tree, q = 1, ...)

## S3 method for class 'numeric'
div_phylo(
  x,
  tree,
  q = 1,
  normalize = TRUE,
 estimator = c("UnveilJ", "ChaoJost", "ChaoShen", "GenCov", "Grassberger", "Marcon",
    "UnveilC", "UnveiliC", "ZhangGrabchak", "naive", "Bonachela", "Holste"),
  level = NULL,
  probability_estimator = c("Chao2015", "Chao2013", "ChaoShen", "naive"),
  unveiling = c("geometric", "uniform", "none"),
  richness_estimator = c("jackknife", "iChao1", "Chao1", "naive"),
  jack_alpha = 0.05,
  jack_max = 10,
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  as_numeric = FALSE,
  ...,
  check_arguments = TRUE
)

## S3 method for class 'species_distribution'
div_phylo(
  x,
  tree,
  q = 1,
  normalize = TRUE,
 estimator = c("UnveilJ", "ChaoJost", "ChaoShen", "GenCov", "Grassberger", "Marcon",
   "UnveilC", "UnveiliC", "ZhangGrabchak", "naive", "Bonachela", "Holste"),
  level = NULL,
  probability_estimator = c("Chao2015", "Chao2013", "ChaoShen", "naive"),
  unveiling = c("geometric", "uniform", "none"),
  richness_estimator = c("jackknife", "iChao1", "Chao1", "naive"),
  jack_alpha = 0.05,
  jack_max = 10,
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  gamma = FALSE,
  ...,
  check_arguments = TRUE
)
```

Arguments

`x`	An object, that may be a numeric vector containing abundances or probabilities, or an object of class abundances or probabilities.
`tree`	an ultrametric, phylogenetic tree. May be an object of class phylo_divent, ape::phylo, ade4::phylog or stats::hclust.
`q`	a number: the order of diversity.
`...`	Unused.
`normalize`	if `TRUE`, phylogenetic is normalized: the height of the tree is set to 1.
`estimator`	An estimator of asymptotic diversity.
`level`	the level of interpolation or extrapolation. It may be a sample size (an integer) or a sample coverage (a number between 0 and 1). If not `NULL`, the asymptotic `estimator` is ignored.
`probability_estimator`	a string containing one of the possible estimators of the probability distribution (see probabilities). Used only for extrapolation.
`unveiling`	a string containing one of the possible unveiling methods to estimate the probabilities of the unobserved species (see probabilities). Used only for extrapolation.

`richness_estimator`	an estimator of richness to evaluate the total number of species, see div_richness. used for interpolation and extrapolation.
`jack_alpha`	the risk level, 5% by default, used to optimize the jackknife order.
`jack_max`	the highest jackknife order allowed. Default is 10.
`coverage_estimator`	an estimator of sample coverage used by coverage.
`as_numeric`	if `TRUE`, a number or a numeric vector is returned rather than a tibble.
`check_arguments`	if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.
`gamma`	if `TRUE`, γ diversity, i.e. diversity of the metacommunity, is computed.

Details

Bias correction requires the number of individuals. See div_hill for estimators.

Entropy can be estimated at a specified level of interpolation or extrapolation, either a chosen sample size or sample coverage (Chao et al. 2014), rather than its asymptotic value. See accum_tsallis for details.

Value

A tibble with the site names, the estimators used and the estimated diversity

References

Chao A, Gotelli NJ, Hsieh TC, Sander EL, Ma KH, Colwell RK, Ellison AM (2014). “Rarefaction and Extrapolation with Hill Numbers: A Framework for Sampling and Estimation in Species Diversity Studies.” *Ecological Monographs*, **84**(1), 45–67. doi:10.1890/130133.1.

Examples

```
div_phylo(paracou_6_abd, tree = paracou_6_taxo, q = 2)

# At 80% coverage
div_phylo(paracou_6_abd, tree = paracou_6_taxo, q = 2, level = 0.8)

# Gamma entropy
div_phylo(paracou_6_abd, tree = paracou_6_taxo, q = 2, gamma = TRUE)
```

div_richness *Number of Species of a Community*

Description

Estimate the number of species from abundance or probability data. Several estimators are available to deal with incomplete sampling.

Usage

```
div_richness(x, ...)

## S3 method for class 'numeric'
div_richness(
  x,
  estimator = c("jackknife", "iChao1", "Chao1", "rarefy", "naive"),
  jack_alpha = 0.05,
  jack_max = 10,
  level = NULL,
  probability_estimator = c("Chao2015", "Chao2013", "ChaoShen", "naive"),
  unveiling = c("geometric", "uniform", "none"),
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  as_numeric = FALSE,
  ...,
  check_arguments = TRUE
)

## S3 method for class 'species_distribution'
div_richness(
  x,
  estimator = c("jackknife", "iChao1", "Chao1", "rarefy", "naive"),
  jack_alpha = 0.05,
  jack_max = 10,
  level = NULL,
  probability_estimator = c("Chao2015", "Chao2013", "ChaoShen", "naive"),
  unveiling = c("geometric", "uniform", "none"),
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  gamma = FALSE,
  as_numeric = FALSE,
  ...,
  check_arguments = TRUE
)
```

Arguments

x	An object, that may be a numeric vector containing abundances or probabilities, or an object of class abundances or probabilities.

`...`	Unused. The metacommunity if built by combining the community abundances with respect to their weight.
`estimator`	An estimator of richness to evaluate the total number of species.
`jack_alpha`	the risk level, 5% by default, used to optimize the jackknife order.
`jack_max`	the highest jackknife order allowed. Default is 10.
`level`	The level of interpolation or extrapolation. It may be a sample size (an integer) or a sample coverage (a number between 0 and 1). The asymptotic `estimator` is used in extrapolation (i.e. a `level` greater than the sample size).
`probability_estimator`	A string containing one of the possible estimators of the probability distribution (see probabilities). Used only by the estimator of richness "rarefy".
`unveiling`	A string containing one of the possible unveiling methods to estimate the probabilities of the unobserved species (see probabilities). Used only by the estimator of richness "rarefy".
`coverage_estimator`	an estimator of sample coverage used by coverage.
`as_numeric`	if `TRUE`, a number or a numeric vector is returned rather than a tibble.
`check_arguments`	if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.
`gamma`	if `TRUE`, γ diversity, i.e. diversity of the metacommunity, is computed.

Details

Bias correction requires the number of individuals. Chao's estimation techniques are from Chao et al. (2014) and Chiu et al. (2014). The Jackknife estimator is calculated by a straight adaptation of the code by Ji-Ping Wang (jackknife in package **SPECIES**). The optimal order is selected according to Burnham and Overton (1978); Burnham and Overton (1979). Many other estimators are available elsewhere, the ones implemented here are necessary for other entropy estimations.

Richness can be estimated at a specified `level` of interpolation or extrapolation, either a chosen sample size or sample coverage (Chiu et al. 2014), rather than its asymptotic value. Extrapolation relies on the estimation of the asymptotic richness. If `probability_estimator` is "naive", then the asymptotic estimation of richness is made using the chosen `estimator`, else the asymptotic distribution of the community is derived and its estimated richness adjusted so that the richness of a sample of this distribution of the size of the actual sample has the richness of the actual sample.

Value

A tibble with the site names, the estimators used and the estimated numbers of species.

References

Burnham KP, Overton WS (1978). "Estimation of the Size of a Closed Population When Capture Probabilities Vary among Animals." *Biometrika*, **65**(3), 625–633. doi:10.2307/2335915.

Burnham KP, Overton WS (1979). “Robust Estimation of Population Size When Capture Probabilities Vary among Animals.” *Ecology*, **60**(5), 927–936. doi:10.2307/1936861.

Chao A, Gotelli NJ, Hsieh TC, Sander EL, Ma KH, Colwell RK, Ellison AM (2014). “Rarefaction and Extrapolation with Hill Numbers: A Framework for Sampling and Estimation in Species Diversity Studies.” *Ecological Monographs*, **84**(1), 45–67. doi:10.1890/130133.1.

Chiu C, Wang Y, Walther BA, Chao A (2014). “An Improved Nonparametric Lower Bound of Species Richness via a Modified Good-Turing Frequency Formula.” *Biometrics*, **70**(3), 671–682. doi:10.1111/biom.12200, 24945937.

Examples

```
# Diversity of each community
div_richness(paracou_6_abd)
# gamma diversity
div_richness(paracou_6_abd, gamma = TRUE)

# At 80% coverage
div_richness(paracou_6_abd, level = 0.8)
```

---

div_similarity *Similarity-Based Diversity of a Community*

Description

Estimate the diversity of species from abundance or probability data and a similarity matrix between species. Several estimators are available to deal with incomplete sampling. Bias correction requires the number of individuals.

Usage

```
div_similarity(x, similarities, q = 1, ...)

## S3 method for class 'numeric'
div_similarity(
  x,
  similarities = diag(length(x)),
  q = 1,
 estimator = c("UnveilJ", "Max", "ChaoShen", "MarconZhang", "UnveilC", "UnveiliC",
    "naive"),
  probability_estimator = c("Chao2015", "Chao2013", "ChaoShen", "naive"),
  unveiling = c("geometric", "uniform", "none"),
  jack_alpha = 0.05,
  jack_max = 10,
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  sample_coverage = NULL,
  as_numeric = FALSE,
  ...,
  check_arguments = TRUE
)

## S3 method for class 'species_distribution'
div_similarity(
  x,
  similarities = diag(sum(!colnames(x) %in% non_species_columns)),
  q = 1,
 estimator = c("UnveilJ", "Max", "ChaoShen", "MarconZhang", "UnveilC", "UnveiliC",
    "naive"),
  probability_estimator = c("Chao2015", "Chao2013", "ChaoShen", "naive"),
  unveiling = c("geometric", "uniform", "none"),
  jack_alpha = 0.05,
  jack_max = 10,
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  gamma = FALSE,
  ...,
  check_arguments = TRUE
)
```

Arguments

`x`	An object, that may be a numeric vector containing abundances or probabilities, or an object of class abundances or probabilities. If it is a numeric vector, then its length must equal the dimensions of the `similarities` matrix: species are assumed to be in the same order.
`similarities`	a similarity matrix, that can be obtained by fun_similarity. Its default value is the identity matrix.
`q`	a number: the order of diversity.
`...`	Unused.
`estimator`	An estimator of asymptotic diversity.
`probability_estimator`	a string containing one of the possible estimators of the probability distribution (see probabilities). Used only for extrapolation.
`unveiling`	a string containing one of the possible unveiling methods to estimate the probabilities of the unobserved species (see probabilities). Used only for extrapolation.
`jack_alpha`	the risk level, 5% by default, used to optimize the jackknife order.
`jack_max`	the highest jackknife order allowed. Default is 10.
`coverage_estimator`	an estimator of sample coverage used by coverage.
`sample_coverage`	the sample coverage of `x` calculated elsewhere. Used to calculate the gamma diversity of meta-communities, see details.

`as_numeric`	if `TRUE`, a number or a numeric vector is returned rather than a tibble.
`check_arguments`	if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.
`gamma`	if `TRUE`, γ diversity, i.e. diversity of the metacommunity, is computed.

Details

All species of the `species_distribution` must be found in the matrix of `similarities` if it is named. If it is not, its size must equal the number of species. Then, the order of species is assumed to be the same as that of the `species_distribution`.

Similarity-Based diversity can't be interpolated of extrapolated as of the state of the art.

Value

A tibble with the site names, the estimators used and the estimated diversity.

Examples

```
# Similarity matrix
Z <- fun_similarity(paracou_6_fundist)
# Diversity of each community
div_similarity(paracou_6_abd, similarities = Z, q = 2)
# gamma diversity
div_similarity(paracou_6_abd, similarities = Z, q = 2, gamma = TRUE)
```

`ent_hurlbert` *Hurlbert Entropy of a Community*

Description

Estimate the Hurlbert entropy (Hurlbert 1971) of species from abundance or probability data. Several estimators are available to deal with incomplete sampling.

Usage

```
ent_hurlbert(x, k = 2, ...)

## S3 method for class 'numeric'
ent_hurlbert(
  x,
  k = 2,
  estimator = c("Hurlbert", "naive"),
  as_numeric = FALSE,
  ...,
  check_arguments = TRUE
)

## S3 method for class 'species_distribution'
ent_hurlbert(
  x,
  k = 2,
  estimator = c("Hurlbert", "naive"),
  as_numeric = FALSE,
  ...,
  check_arguments = TRUE
)
```

Arguments

`x`	An object, that may be a numeric vector containing abundances or probabilities, or an object of class abundances or probabilities.
`k`	the order of Hurlbert's diversity.
`...`	Unused.
`estimator`	An estimator of entropy.
`as_numeric`	if `TRUE`, a number or a numeric vector is returned rather than a tibble.
`check_arguments`	if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

Bias correction requires the number of individuals. See div_hurlbert for estimators.

Hurlbert's entropy cannot be estimated at a specified level of interpolation or extrapolation, and entropy partitioning is not available.

Value

A tibble with the site names, the estimators used and the estimated entropy.

References

Hurlbert SH (1971). "The Nonconcept of Species Diversity: A Critique and Alternative Parameters." *Ecology*, **52**(4), 577–586. doi:10.2307/1934145.

Examples

```
# Entropy of each community
ent_hurlbert(paracou_6_abd, k = 2)
```

`ent_phylo` *Phylogenetic Entropy of a Community*

Description

Estimate the entropy of species from abundance or probability data and a phylogenetic tree. Several estimators are available to deal with incomplete sampling.

Usage

```
ent_phylo(x, tree, q = 1, ...)

## S3 method for class 'numeric'
ent_phylo(
  x,
  tree,
  q = 1,
  normalize = TRUE,
 estimator = c("UnveilJ", "ChaoJost", "ChaoShen", "GenCov", "Grassberger", "Marcon",
    "UnveilC", "UnveiliC", "ZhangGrabchak", "naive", "Bonachela", "Holste"),
  level = NULL,
  probability_estimator = c("Chao2015", "Chao2013", "ChaoShen", "naive"),
  unveiling = c("geometric", "uniform", "none"),
  richness_estimator = c("jackknife", "iChao1", "Chao1", "naive"),
  jack_alpha = 0.05,
  jack_max = 10,
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  as_numeric = FALSE,
  ...,
  check_arguments = TRUE
)

## S3 method for class 'species_distribution'
ent_phylo(
  x,
  tree,
  q = 1,
  normalize = TRUE,
 estimator = c("UnveilJ", "ChaoJost", "ChaoShen", "GenCov", "Grassberger", "Marcon",
    "UnveilC", "UnveiliC", "ZhangGrabchak", "naive", "Bonachela", "Holste"),
  level = NULL,
  probability_estimator = c("Chao2015", "Chao2013", "ChaoShen", "naive"),
  unveiling = c("geometric", "uniform", "none"),
  richness_estimator = c("jackknife", "iChao1", "Chao1", "naive"),
  jack_alpha = 0.05,
  jack_max = 10,
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  gamma = FALSE,
  ...,
  check_arguments = TRUE
)
```

Arguments

`x`	An object, that may be a numeric vector containing abundances or probabilities, or an object of class abundances or probabilities.
`tree`	an ultrametric, phylogenetic tree. May be an object of class phylo_divent, ape::phylo, ade4::phylog or stats::hclust.
`q`	a number: the order of diversity.
`...`	Unused.
`normalize`	if `TRUE`, phylogenetic is normalized: the height of the tree is set to 1.
`estimator`	An estimator of entropy.
`level`	the level of interpolation or extrapolation. It may be a sample size (an integer) or a sample coverage (a number between 0 and 1). If not `NULL`, the asymptotic `estimator` is ignored.
`probability_estimator`	a string containing one of the possible estimators of the probability distribution (see probabilities). Used only for extrapolation.
`unveiling`	a string containing one of the possible unveiling methods to estimate the probabilities of the unobserved species (see probabilities). Used only for extrapolation.
`richness_estimator`	an estimator of richness to evaluate the total number of species, see div_richness. used for interpolation and extrapolation.
`jack_alpha`	the risk level, 5% by default, used to optimize the jackknife order.
`jack_max`	the highest jackknife order allowed. Default is 10.
`coverage_estimator`	an estimator of sample coverage used by coverage.
`as_numeric`	if `TRUE`, a number or a numeric vector is returned rather than a tibble.
`check_arguments`	if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.
`gamma`	if `TRUE`, γ diversity, i.e. diversity of the metacommunity, is computed.

Details

Bias correction requires the number of individuals. See div_hill for estimators.

Entropy can be estimated at a specified level of interpolation or extrapolation, either a chosen sample size or sample coverage (Chao et al. 2014), rather than its asymptotic value. See accum_tsallis for details.

Value

A tibble with the site names, the estimators used and the estimated entropy.

References

Chao A, Gotelli NJ, Hsieh TC, Sander EL, Ma KH, Colwell RK, Ellison AM (2014). "Rarefaction and Extrapolation with Hill Numbers: A Framework for Sampling and Estimation in Species Diversity Studies." *Ecological Monographs*, **84**(1), 45–67. doi:10.1890/130133.1.

Examples

```
# Entropy of each community
ent_phylo(paracou_6_abd, tree = paracou_6_taxo, q = 2)
# Gamma entropy
ent_phylo(paracou_6_abd, tree = paracou_6_taxo, q = 2, gamma = TRUE)

# At 80% coverage
ent_phylo(paracou_6_abd, tree = paracou_6_taxo, q = 2, level = 0.8)
```

ent_rao *Rao's Quadratic Entropy of a Community*

Description

Estimate the quadratic entropy (Rao 1982) of species from abundance or probability data. An estimator (Lande 1996) is available to deal with incomplete sampling.

Usage

```
ent_rao(x, ...)

## S3 method for class 'numeric'
ent_rao(
  x,
  distances = NULL,
  tree = NULL,
  normalize = TRUE,
  estimator = c("Lande", "naive"),
  as_numeric = FALSE,
  ...,
  check_arguments = TRUE
)

## S3 method for class 'species_distribution'
ent_rao(
  x,
  distances = NULL,
  tree = NULL,
  normalize = TRUE,
  estimator = c("Lande", "naive"),
  gamma = FALSE,
  ...,
  check_arguments = TRUE
)
```

Arguments

`x`	An object, that may be a numeric vector containing abundances or probabilities, or an object of class abundances or probabilities.
`...`	Unused.
`distances`	a distance matrix or an object of class stats::dist.
`tree`	an ultrametric, phylogenetic tree. May be an object of class phylo_divent, ape::phylo, ade4::phylog or stats::hclust.
`normalize`	if `TRUE`, phylogenetic is normalized: the height of the tree is set to 1.
`estimator`	An estimator of entropy.
`as_numeric`	if `TRUE`, a number or a numeric vector is returned rather than a tibble.
`check_arguments`	if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.
`gamma`	if `TRUE`, γ diversity, i.e. diversity of the metacommunity, is computed.

Details

Rao's entropy is phylogenetic or similarity-based entropy of order 2. ent_phylo and ent_similarity with argument `q = 2` provide more estimators and allow estimating entropy at a chosen level.

All species of the `species_distribution` must be found in the matrix of `distances` if it is named. If it is not or if `x` is numeric, its size must equal the number of species. Then, the order of species is assumed to be the same as that of the `species_distribution` or its numeric equivalent.

Value

A tibble with the site names, the estimators used and the estimated entropy.

References

Lande R (1996). "Statistics and Partitioning of Species Diversity, and Similarity among Multiple Communities." *Oikos*, **76**(1), 5–13. doi:10.2307/3545743.

Rao CR (1982). "Diversity and Dissimilarity Coefficients: A Unified Approach." *Theoretical Population Biology*, **21**, 24–43. doi:10.1016/00405809(82)900041.

Examples

```
# Entropy of each community
ent_rao(paracou_6_abd, tree = paracou_6_taxo)
# Similar to (but estimators are not the same)
ent_phylo(paracou_6_abd, tree = paracou_6_taxo, q = 2)

# Functional entropy
ent_rao(paracou_6_abd, distances = paracou_6_fundist)

# gamma entropy
ent_rao(paracou_6_abd, tree = paracou_6_taxo, gamma = TRUE)
```

`ent_shannon` *Shannon's Entropy of a Community*

Description

Estimate the entropy (Shannon 1948) of species from abundance or probability data. Several estimators are available to deal with incomplete sampling.

Usage

```
ent_shannon(x, ...)

## S3 method for class 'numeric'
ent_shannon(
  x,
 estimator = c("UnveilJ", "ChaoJost", "ChaoShen", "GenCov", "Grassberger", "Marcon",
   "UnveilC", "UnveiliC", "ZhangGrabchak", "naive", "Bonachela", "Grassberger2003",
    "Holste", "Miller", "Schurmann", "ZhangHz"),
  level = NULL,
  probability_estimator = c("Chao2015", "Chao2013", "ChaoShen", "naive"),
  unveiling = c("geometric", "uniform", "none"),
  richness_estimator = c("jackknife", "iChao1", "Chao1", "naive"),
  jack_alpha = 0.05,
  jack_max = 10,
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  as_numeric = FALSE,
  ...,
  check_arguments = TRUE
)

## S3 method for class 'species_distribution'
ent_shannon(
  x,
 estimator = c("UnveilJ", "ChaoJost", "ChaoShen", "GenCov", "Grassberger", "Marcon",
    "UnveilC", "UnveiliC", "ZhangGrabchak", "naive", "Bonachela", "Grassberger2003",
      "Holste", "Miller", "Schurmann", "ZhangHz"),
  level = NULL,
  probability_estimator = c("Chao2015", "Chao2013", "ChaoShen", "naive"),
  unveiling = c("geometric", "uniform", "none"),
  richness_estimator = c("jackknife", "iChao1", "Chao1", "naive"),
  jack_alpha = 0.05,
  jack_max = 10,
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  gamma = FALSE,
  as_numeric = FALSE,
  ...,
  check_arguments = TRUE
)
```

Arguments

`x`	An object, that may be a numeric vector containing abundances or probabilities, or an object of class abundances or probabilities.
`...`	Unused.
`estimator`	An estimator of entropy.
`level`	the level of interpolation or extrapolation. It may be a sample size (an integer) or a sample coverage (a number between 0 and 1). If not `NULL`, the asymptotic `estimator` is ignored.
`probability_estimator`	a string containing one of the possible estimators of the probability distribution (see probabilities). Used only for extrapolation.
`unveiling`	a string containing one of the possible unveiling methods to estimate the probabilities of the unobserved species (see probabilities). Used only for extrapolation.
`richness_estimator`	an estimator of richness to evaluate the total number of species, see div_richness. used for interpolation and extrapolation.
`jack_alpha`	the risk level, 5% by default, used to optimize the jackknife order.
`jack_max`	the highest jackknife order allowed. Default is 10.
`coverage_estimator`	an estimator of sample coverage used by coverage.
`as_numeric`	if `TRUE`, a number or a numeric vector is returned rather than a tibble.
`check_arguments`	if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.
`gamma`	if `TRUE`, γ diversity, i.e. diversity of the metacommunity, is computed.

Details

Bias correction requires the number of individuals.

See div_hill for non-specific estimators. Shannon-specific estimators are from Miller (1955), Grassberger (2003), Schürmann (2004) and Zhang (2012). More estimators can be found in the **entropy** package.

Entropy can be estimated at a specified level of interpolation or extrapolation, either a chosen sample size or sample coverage (Chao et al. 2014), rather than its asymptotic value. See accum_tsallis for details.

Value

A tibble with the site names, the estimators used and the estimated entropy.

References

Chao A, Gotelli NJ, Hsieh TC, Sander EL, Ma KH, Colwell RK, Ellison AM (2014). "Rarefaction and Extrapolation with Hill Numbers: A Framework for Sampling and Estimation in Species Diversity Studies." *Ecological Monographs*, **84**(1), 45–67. doi:10.1890/130133.1.

Grassberger P (2003). "Entropy Estimates from Insufficient Samplings." *arXiv Physics e-prints*, **0307138**(v2).

Miller GA (1955). "Note on the Bias of Information Estimates." In Quastler H (ed.), *Information Theory in Psychology: Problems and Methods*, 95–100. Free Press, Glencoe, Ill.

Schürmann T (2004). "Bias Analysis in Entropy Estimation." *Journal of Physics A: Mathematical and General*, **37**(27), L295–L301. doi:10.1088/03054470/37/27/L02.

Shannon CE (1948). "A Mathematical Theory of Communication." *The Bell System Technical Journal*, **27**(3), 379–423, 623–656. doi:10.1002/j.15387305.1948.tb01338.x.

Zhang Z (2012). "Entropy Estimation in Turing's Perspective." *Neural Computation*, **24**(5), 1368–1389. doi:10.1162/NECO_a_00266.

Examples

```
# Entropy of each community
ent_shannon(paracou_6_abd)
# gamma entropy
ent_shannon(paracou_6_abd, gamma = TRUE)

# At 80% coverage
ent_shannon(paracou_6_abd, level = 0.8)
```

ent_similarity *Similarity-Based Entropy of a Community*

Description

Estimate the entropy of species from abundance or probability data and a similarity matrix between species. Several estimators are available to deal with incomplete sampling. Bias correction requires the number of individuals.

Usage

```
ent_similarity(x, similarities, q = 1, ...)

## S3 method for class 'numeric'
ent_similarity(
  x,
  similarities = diag(length(x)),
  q = 1,
  estimator = c("UnveilJ", "Max", "ChaoShen", "MarconZhang", "UnveilC", "UnveiliC",
    "naive"),
  probability_estimator = c("Chao2015", "Chao2013", "ChaoShen", "naive"),
  unveiling = c("geometric", "uniform", "none"),
  jack_alpha = 0.05,
  jack_max = 10,
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  sample_coverage = NULL,
  as_numeric = FALSE,
  ...,
  check_arguments = TRUE
)

## S3 method for class 'species_distribution'
ent_similarity(
  x,
  similarities = diag(sum(!colnames(x) %in% non_species_columns)),
  q = 1,
  estimator = c("UnveilJ", "Max", "ChaoShen", "MarconZhang", "UnveilC", "UnveiliC",
    "naive"),
  probability_estimator = c("Chao2015", "Chao2013", "ChaoShen", "naive"),
  unveiling = c("geometric", "uniform", "none"),
  jack_alpha = 0.05,
  jack_max = 10,
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  gamma = FALSE,
  ...,
  check_arguments = TRUE
)
```

Arguments

`x`	An object, that may be a numeric vector containing abundances or probabilities, or an object of class abundances or probabilities. If it is a numeric vector, then its length must equal the dimensions of the `similarities` matrix: species are assumed to be in the same order.
`similarities`	a similarity matrix, that can be obtained by fun_similarity. Its default value is the identity matrix.
`q`	a number: the order of diversity.
`...`	Unused.
`estimator`	An estimator of entropy.
`probability_estimator`	a string containing one of the possible estimators of the probability distribution (see probabilities). Used only for extrapolation.
`unveiling`	a string containing one of the possible unveiling methods to estimate the probabilities of the unobserved species (see probabilities). Used only for extrapolation.
`jack_alpha`	the risk level, 5% by default, used to optimize the jackknife order.
`jack_max`	the highest jackknife order allowed. Default is 10.
`coverage_estimator`	an estimator of sample coverage used by coverage.
`sample_coverage`	the sample coverage of `x` calculated elsewhere. Used to calculate the gamma diversity of meta-communities, see details.
`as_numeric`	if `TRUE`, a number or a numeric vector is returned rather than a tibble.
`check_arguments`	if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.
`gamma`	if `TRUE`, γ diversity, i.e. diversity of the metacommunity, is computed.

Details

All species of the `species_distribution` must be found in the matrix of `similarities` if it is named. If it is not or if `x` is numeric, its size must equal the number of species. Then, the order of species is assumed to be the same as that of the `species_distribution` or its numeric equivalent.

Similarity-Based entropy can't be interpolated of extrapolated as of the state of the art.

Value

A tibble with the site names, the estimators used and the estimated entropy.

Examples

```
# Similarity matrix
Z <- fun_similarity(paracou_6_fundist)
# Diversity of each community
```

```
ent_similarity(paracou_6_abd, similarities = Z, q = 2)
# gamma diversity
ent_similarity(paracou_6_abd, similarities = Z, q = 2, gamma = TRUE)
```

`ent_simpson` *Simpson's Entropy of a Community*

Description

Estimate the entropy (Simpson 1949) of species from abundance or probability data. Several estimators are available to deal with incomplete sampling.

Usage

```
ent_simpson(x, ...)

## S3 method for class 'numeric'
ent_simpson(
  x,
 estimator = c("Lande", "UnveilJ", "ChaoJost", "ChaoShen", "GenCov", "Grassberger",
   "Marcon", "UnveilC", "UnveiliC", "ZhangGrabchak", "naive", "Bonachela", "Holste"),
  level = NULL,
  probability_estimator = c("Chao2015", "Chao2013", "ChaoShen", "naive"),
  unveiling = c("geometric", "uniform", "none"),
  richness_estimator = c("jackknife", "iChao1", "Chao1", "naive"),
  jack_alpha = 0.05,
  jack_max = 10,
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  as_numeric = FALSE,
  ...,
  check_arguments = TRUE
)

## S3 method for class 'species_distribution'
ent_simpson(
  x,
 estimator = c("Lande", "UnveilJ", "ChaoJost", "ChaoShen", "GenCov", "Grassberger",
   "Marcon", "UnveilC", "UnveiliC", "ZhangGrabchak", "naive", "Bonachela", "Holste"),
  level = NULL,
  probability_estimator = c("Chao2015", "Chao2013", "ChaoShen", "naive"),
  unveiling = c("geometric", "uniform", "none"),
  richness_estimator = c("jackknife", "iChao1", "Chao1", "naive"),
  jack_alpha = 0.05,
  jack_max = 10,
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  gamma = FALSE,
  as_numeric = FALSE,
  ...,
  check_arguments = TRUE
)
```

Arguments

`x`	An object, that may be a numeric vector containing abundances or probabilities, or an object of class abundances or probabilities.
`...`	Unused.
`estimator`	An estimator of entropy.
`level`	the level of interpolation or extrapolation. It may be a sample size (an integer) or a sample coverage (a number between 0 and 1). If not `NULL`, the asymptotic `estimator` is ignored.
`probability_estimator`	a string containing one of the possible estimators of the probability distribution (see probabilities). Used only for extrapolation.
`unveiling`	a string containing one of the possible unveiling methods to estimate the probabilities of the unobserved species (see probabilities). Used only for extrapolation.
`richness_estimator`	an estimator of richness to evaluate the total number of species, see div_richness. used for interpolation and extrapolation.
`jack_alpha`	the risk level, 5% by default, used to optimize the jackknife order.
`jack_max`	the highest jackknife order allowed. Default is 10.
`coverage_estimator`	an estimator of sample coverage used by coverage.
`as_numeric`	if `TRUE`, a number or a numeric vector is returned rather than a tibble.
`check_arguments`	if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.
`gamma`	if `TRUE`, γ diversity, i.e. diversity of the metacommunity, is computed.

Details

Bias correction requires the number of individuals. See div_hill for non-specific estimators.

Simpson-specific estimator is from Lande (1996).

Entropy can be estimated at a specified level of interpolation or extrapolation, either a chosen sample size or sample coverage (Chao et al. 2014), rather than its asymptotic value. See accum_tsallis for details.

Value

A tibble with the site names, the estimators used and the estimated entropy.

References

Chao A, Gotelli NJ, Hsieh TC, Sander EL, Ma KH, Colwell RK, Ellison AM (2014). “Rarefaction and Extrapolation with Hill Numbers: A Framework for Sampling and Estimation in Species Diversity Studies.” *Ecological Monographs*, **84**(1), 45–67. doi:10.1890/130133.1.

Lande R (1996). “Statistics and Partitioning of Species Diversity, and Similarity among Multiple Communities.” *Oikos*, **76**(1), 5–13. doi:10.2307/3545743.

Simpson EH (1949). “Measurement of Diversity.” *Nature*, **163**(4148), 688. doi:10.1038/163688a0.

Examples

```
# Entropy of each community
ent_simpson(paracou_6_abd)
# gamma entropy
ent_simpson(paracou_6_abd, gamma = TRUE)

# At 80% coverage
ent_simpson(paracou_6_abd, level = 0.8)
```

`ent_sp_simpson` *Spatially Explicit Simpson’s Entropy*

Description

Simpson’s entropy of the neighborhood of individuals, up to a distance (Shimatani 2001).

Usage

```
ent_sp_simpson(
  X,
  r = NULL,
  correction = c("isotropic", "translate", "none"),
  check_arguments = TRUE
)

ent_sp_simpsonEnvelope(
  X,
  r = NULL,
  n_simulations = 100,
  alpha = 0.05,
  correction = c("isotropic", "translate", "none"),
  h0 = c("RandomPosition", "RandomLabeling"),
  global = FALSE,
  check_arguments = TRUE
)
```

Arguments

`X`	a spatialized community (A dbmss::wmppp object with `PointType` values as species names.)
`r`	a vector of distances.
`correction`	the edge-effect correction to apply when estimating the number of neighbors or the *K* function with spatstat.explore::Kest. Default is "isotropic".
`check_arguments`	if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.
`n_simulations`	the number of simulations used to estimate the confidence envelope.
`alpha`	the risk level, 5% by default.
`h0`	A string describing the null hypothesis to simulate. The null hypothesis may be "RandomPosition": points are drawn in a Poisson process (default) or "RandomLabeling": randomizes point types, keeping locations unchanged.
`global`	if `TRUE`, a global envelope sensu (Duranton and Overman 2005) is calculated.

Value

`ent_sp_simpson` returns an object of class `fv`, see spatstat.explore::fv.object. There are methods to print and plot this class. It contains the value of the spatially explicit Simpson's entropy for each distance in `r`.

`ent_sp_simpsonEnvelope` returns an envelope object spatstat.explore::envelope. There are methods to print and plot this class. It contains the observed value of the function, its average simulated value and the confidence envelope.

References

Duranton G, Overman HG (2005). "Testing for Localisation Using Micro-Geographic Data." *Review of Economic Studies*, **72**(4), 1077–1106. doi:10.1111/00346527.00362.

Shimatani K (2001). "Multivariate Point Processes and Spatial Variation of Species Diversity." *Forest Ecology and Management*, **142**(1-3), 215–229. doi:10.1016/s03781127(00)003522.

Examples

```
# Generate a random community
X <- rspcommunity(1, size = 1000, species_number = 3)
# Calculate the entropy and plot it
autoplot(ent_sp_simpson(X))

# Generate a random community
X <- rspcommunity(1, size = 1000, species_number = 3)
# Calculate the entropy and plot it
autoplot(ent_sp_simpsonEnvelope(X, n_simulations = 10))
```

ent_tsallis *Tsallis Entropy of a Community*

Description

Estimate the entropy of species from abundance or probability data. Several estimators are available to deal with incomplete sampling.

Usage

```
ent_tsallis(x, q = 1, ...)

## S3 method for class 'numeric'
ent_tsallis(
  x,
  q = 1,
 estimator = c("UnveilJ", "ChaoJost", "ChaoShen", "GenCov", "Grassberger", "Marcon",
    "UnveilC", "UnveiliC", "ZhangGrabchak", "naive", "Bonachela", "Holste"),
  level = NULL,
  probability_estimator = c("Chao2015", "Chao2013", "ChaoShen", "naive"),
  unveiling = c("geometric", "uniform", "none"),
  richness_estimator = c("jackknife", "iChao1", "Chao1", "rarefy", "naive"),
  jack_alpha = 0.05,
  jack_max = 10,
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  sample_coverage = NULL,
  as_numeric = FALSE,
  ...,
  check_arguments = TRUE
)

## S3 method for class 'species_distribution'
ent_tsallis(
  x,
  q = 1,
 estimator = c("UnveilJ", "ChaoJost", "ChaoShen", "GenCov", "Grassberger", "Marcon",
    "UnveilC", "UnveiliC", "ZhangGrabchak", "naive", "Bonachela", "Holste"),
  level = NULL,
  probability_estimator = c("Chao2015", "Chao2013", "ChaoShen", "naive"),
  unveiling = c("geometric", "uniform", "none"),
  richness_estimator = c("jackknife", "iChao1", "Chao1", "rarefy", "naive"),
  jack_alpha = 0.05,
  jack_max = 10,
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  gamma = FALSE,
  as_numeric = FALSE,
  ...,
  check_arguments = TRUE
)
```

Arguments

`x`	An object, that may be a numeric vector containing abundances or probabilities, or an object of class abundances or probabilities.
`q`	a number: the order of diversity.
`...`	Unused.
`estimator`	An estimator of entropy.
`level`	the level of interpolation or extrapolation. It may be a sample size (an integer) or a sample coverage (a number between 0 and 1). If not `NULL`, the asymptotic `estimator` is ignored.
`probability_estimator`	a string containing one of the possible estimators of the probability distribution (see probabilities). Used only for extrapolation.
`unveiling`	a string containing one of the possible unveiling methods to estimate the probabilities of the unobserved species (see probabilities). Used only for extrapolation.
`richness_estimator`	an estimator of richness to evaluate the total number of species, see div_richness. used for interpolation and extrapolation.
`jack_alpha`	the risk level, 5% by default, used to optimize the jackknife order.
`jack_max`	the highest jackknife order allowed. Default is 10.
`coverage_estimator`	an estimator of sample coverage used by coverage.
`sample_coverage`	the sample coverage of `x` calculated elsewhere. Used to calculate the gamma diversity of meta-communities, see details.
`as_numeric`	if `TRUE`, a number or a numeric vector is returned rather than a tibble.
`check_arguments`	if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.
`gamma`	if `TRUE`, γ diversity, i.e. diversity of the metacommunity, is computed.

Details

Bias correction requires the number of individuals. See div_hill for estimators.

Entropy can be estimated at a specified level of interpolation or extrapolation, either a chosen sample size or sample coverage (Chao et al. 2014), rather than its asymptotic value. See accum_tsallis for details.

Value

A tibble with the site names, the estimators used and the estimated entropy.

References

Chao A, Gotelli NJ, Hsieh TC, Sander EL, Ma KH, Colwell RK, Ellison AM (2014). "Rarefaction and Extrapolation with Hill Numbers: A Framework for Sampling and Estimation in Species Diversity Studies." *Ecological Monographs*, **84**(1), 45–67. doi:10.1890/130133.1.

Examples

```
# Entropy of each community
ent_tsallis(paracou_6_abd, q = 2)
# gamma entropy
ent_tsallis(paracou_6_abd, q = 2, gamma = TRUE)

# At 80% coverage
ent_tsallis(paracou_6_abd, level = 0.8)
```

---

exp_q *Deformed exponential*

Description

Calculate the deformed exponential of order q.

Usage

```
exp_q(x, q)
```

Arguments

x	A numeric vector or array.
q	A number.

Details

The deformed exponential is the reciprocal of the deformed logarithm (Tsallis 1994), see ln_q. It is defined as $(x(1-q)+1)^{\frac{1}{(1-q)}}$. For $q > 1$, $\ln_q(+\infty) = \frac{1}{(q-1)}$ so $\exp_q(x)$ is not defined for $x > \frac{1}{(q-1)}$.

Value

A vector of the same length as x containing the transformed values.

References

Tsallis C (1994). "What Are the Numbers That Experiments Provide?" *Química Nova*, **17**(6), 468–471.

Examples

```
curve(exp_q(x, q = 0), from = -5, to = 0, lty = 2)
curve(exp(x), from = -5, to = 0, lty= 1, add = TRUE)
curve(exp_q(x, q = 2), from = -5, to = 0, lty = 3, add = TRUE)
legend("bottomright",
  legend = c(
    expression(exp[0](x)),
    expression(exp(x)),
    expression(exp[2](x))
  ),
  lty = c(2, 1, 3),
  inset = 0.02
)
```

---

e_n_q *Grassberger's expectation of n^q*

Description

Expected value of n^q when n follows a Poisson distribution of parameter n.

Usage

```
e_n_q(n, q)
```

Arguments

n	A positive integer vector.
q	A positive number.

Details

The expectation of n^q when n follows a Poisson distribution was derived by Grassberger (1988).

It is computed using the beta function. Its value is 0 for $n - q + 1 < 0$, and close to 0 when $n = q$, which is not a correct estimate: it should not be used when $q > n$.

Value

A vector of the same length as n containing the transformed values.

References

Grassberger P (1988). "Finite Sample Corrections to Entropy and Dimension Estimates." *Physics Letters A*, **128**(6-7), 369–373. doi:10.1016/03759601(88)901934.

Examples

```
n <- 10
q <- 2
# Compare
e_n_q(n, q)
# with (empirical estimation)
mean(rpois(1000, lambda = n)^q)
# and (naive estimation)
n^q
```

---

fit_rac *Fit a distribution*

Description

Fit a well-known distribution to a species distribution.

Usage

```
fit_rac(x, ...)

## S3 method for class 'numeric'
fit_rac(
  x,
  distribution = c("lnorm", "lseries", "geom", "bstick"),
  ...,
  check_arguments = TRUE
)

## S3 method for class 'species_distribution'
fit_rac(
  x,
  distribution = c("lnorm", "lseries", "geom", "bstick"),
  ...,
  check_arguments = TRUE
)
```

Arguments

`x`	An object
`...`	Unused.
`distribution`	The distribution of species abundances. May be "lnorm" (log-normal), "lseries" (log-series), "geom" (geometric) or "bstick" (broken stick).
`check_arguments`	if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

abundances can be used to fit rank-abundance curves (RAC) of classical distributions:

- "lnorm" for log-normal (Preston 1948).
- "lseries" for log-series (Fisher et al. 1943).
- "geom" for geometric (Motomura 1932).
- "bstick" for broken stick (MacArthur 1957). It has no parameter, so the maximum abundance is returned.

Value

A tibble with the sites and the estimated distribution parameters.

References

Fisher RA, Corbet AS, Williams CB (1943). "The Relation between the Number of Species and the Number of Individuals in a Random Sample of an Animal Population." *Journal of Animal Ecology*, **12**, 42–58. doi:10.2307/1411.

MacArthur RH (1957). "On the Relative Abundance of Bird Species." *Proceedings of the National Academy of Sciences of the United States of America*, **43**(3), 293–295. doi:10.1073/pnas.43.3.293, 89566.

Motomura I (1932). "On the statistical treatment of communities." *Zoological Magazine*, **44**, 379–383.

Preston FW (1948). "The Commonness, and Rarity, of Species." *Ecology*, **29**(3), 254–283. doi:10.2307/1930989.

Examples

```
fit_rac(paracou_6_abd, distribution = "lnorm")
```

`fun_ordinariness` *Functional ordinariness of a community*

Description

The ordinariness of a species is the average similarity of its individuals with others (Leinster and Cobbold 2012).

Usage

```
fun_ordinariness(
  species_distribution,
 similarities = diag(sum(!colnames(species_distribution) %in% non_species_columns)),
  as_numeric = FALSE,
  check_arguments = TRUE
)
```

Arguments

`species_distribution`	an object of class species_distribution.
`similarities`	a similarity matrix, that can be obtained by fun_similarity. Its default value is the identity matrix.
`as_numeric`	if `TRUE`, a number or a numeric vector is returned rather than a tibble.
`check_arguments`	if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

All species of the `species_distribution` must be found in the matrix of `similarities` if it is named. If it is not, its size must equal the number of species. Then, the order of species is assumed to be the same as that of the `species_distribution`.

Value

A tibble with the ordinariness of each species, or a matrix if argument `as_numeric` is `TRUE`.

References

Leinster T, Cobbold C (2012). “Measuring Diversity: The Importance of Species Similarity.” *Ecology*, **93**(3), 477–489. doi:10.1890/102402.1.

Examples

```
fun_ordinariness(paracou_6_abd, fun_similarity(paracou_6_fundist))

# Compare with probabilities
probabilities(paracou_6_abd)
# Decrease similarities so that ordinariness is close to probability
fun_ordinariness(paracou_6_abd, fun_similarity(paracou_6_fundist, rate = 100))
```

fun_similarity *Functional similarity*

Description

Transform a distance matrix into a similarity matrix (Leinster and Cobbold 2012). Similarity between two species is defined either by a negative exponential function of their distance or by the complement to 1 of their normalized distance (such that the most distant species are 1 apart).

Usage

```
fun_similarity(distances, exponential = TRUE, rate = 1, check_arguments = TRUE)
```

Arguments

`distances`	a distance matrix or an object of class stats::dist.
`exponential`	If `TRUE`, similarity is $e^{-r\delta}$, where r is argument `rate`. If `FALSE`, it is $1 - \delta/\max(\delta)$.
`rate`	the decay rate of the exponential similarity.
`check_arguments`	if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Value

A similarity matrix.

References

Leinster T, Cobbold C (2012). “Measuring Diversity: The Importance of Species Similarity.” *Ecology*, **93**(3), 477–489. doi:10.1890/102402.1.

Examples

```
# Similarity between Paracou 6 species
hist(fun_similarity(paracou_6_fundist))
```

ln_q *Deformed logarithm*

Description

Calculate the deformed logarithm of order q.

Usage

```
ln_q(x, q)
```

Arguments

x	A numeric vector or array.
q	A number.

Details

The deformed logarithm (Tsallis 1994) is defined as $\ln_q x = \frac{(x^{(1-q)} - 1)}{(1-q)}$.

The shape of the deformed logarithm is similar to that of the regular one. $\ln_1 x = \log x$.

For $q > 1$, $\ln_q (+\infty) = \frac{1}{(q-1)}$.

Value

A vector of the same length as x containing the transformed values.

References

Tsallis C (1994). "What Are the Numbers That Experiments Provide?" *Química Nova*, **17**(6), 468–471.

Examples

```
curve(ln_q(1/ x, q = 0), 0, 1, lty = 2, ylab = "Logarithm", ylim = c(0, 10))
curve(log(1 / x), 0, 1, lty = 1, n =1E4, add = TRUE)
curve(ln_q(1 / x, q = 2), 0, 1, lty = 3, add = TRUE)
legend("topright",
  legend = c(
    expression(ln[0](1/x)),
    expression(log(1/x)),
    expression(ln[2](1/x))
  ),
  lty = c(2, 1, 3),
  inset = 0.02
 )
```

metacommunity — *Aggregate communities into a metacommunity*

Description

Abundances of communities are summed according to their weights to obtain the abundances of the metacommunity.

Usage

```
metacommunity(x, name = "metacommunity", ...)

## S3 method for class 'matrix'
metacommunity(
  x,
  name = "metacommunity",
  weights = rep(1, nrow(x)),
  as_numeric = TRUE,
  ...,
  check_arguments = TRUE
)

## S3 method for class 'abundances'
metacommunity(
  x,
  name = "metacommunity",
  as_numeric = FALSE,
  ...,
  check_arguments = TRUE
)
```

Arguments

`x`	An object of class abundances that contains several communities or a matrix of abundances with communities in rows and species in columns.
`name`	The name of the metacommunity
`...`	Unused.
`weights`	the weights of the sites of the species distributions.
`as_numeric`	if `TRUE`, a number or a numeric vector is returned rather than a tibble.
`check_arguments`	if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

The total abundance of the metacommunity is by design equal to the sum of community abundances so that the information used by diversity estimators. A consequence is that equal weights lead to a metacommunity whose species abundances are the sum of community species abundances.

If community weights are not equal then the metacommunity abundances are in general not integer. Most diversity estimators can't be applied to non-integer abundances but the knowledge of the sample coverage of each community allow "ChaoShen" and "Grassberger" estimators.

Value

An object of class abundances with a single row or a named vector if `as_numeric = TRUE`.

Examples

```
metacommunity(paracou_6_abd)

metacommunity(paracou_6_abd)
```

`mock_3sp` *Mock data*

Description

A simple dataset to test diversity functions. It contains 3 species with their abundances, their distance matrix and their phylogenetic tree.

Usage

```
mock_3sp_abd

mock_3sp_dist

mock_3sp_tree
```

Format

`mock_3sp_abd` is a vector, `mock_3sp_dist` a matrix and `mock_3sp_tree` an object of class ape::phylo.

An object of class `dist` of length 3.

An object of class `phylo` of length 4.

Examples

```
mock_3sp_abd
mock_3sp_dist
plot(mock_3sp_tree)
axis(2)
```

paracou_6 *Paracou plot 6*

Description

A community assembly. It contains number of trees per species of the plot #6 of Paracou. The plot covers 6.25 ha of tropical rainforest, divided into 4 equally-sized subplots.

Usage

```
paracou_6_abd

paracou_6_wmppp
```

Format

`paracou_6_abd` is an object of class abundances, which is also a tibble::tibble. Each line of the tibble is a subplot. `paracou_6_wmppp` is a dbmss::wmppp object, i.e. a weighted, marked planar point pattern.

An object of class `wmppp` (inherits from `ppp`) of length 6.

Details

In `paracou_6_abd` (a tibble), the "site" column contains the subplot number, "weight" contains its area and all others columns contain a species. Data are the number of trees above 10 cm diameter at breast height (DBH).

In `paracou_6_wmppp` (a point pattern), the point type is tree species and the point weight is their basal area, in square centimeters.

This dataset is from Paracou field station, French Guiana, managed by Cirad.

Source

Permanent data census of Paracou: `https://paracou.cirad.fr/`

See Also

paracou_6_taxo, paracou_6_fundist

`paracou_6_fundist` *Functional distances between Paracou plot 6 species*

Description

A functional distance matrix of species of the dataset paracou_6_abd. Distances were computed from a trait dataset including specific leaf area, wood density, seed mass and 95th percentile of height. Gower's metric (Gower 1971) was used to obtain a distance matrix.

Usage

```
paracou_6_fundist
```

Format

A matrix.

Details

This dataset is from Paracou field station, French Guiana, managed by Cirad.

Source

Permanent data census of Paracou: `https://paracou.cirad.fr/`

References

Gower JC (1971). "A General Coefficient of Similarity and Some of Its Properties." *Biometrics*, **27**(4), 857–871. doi:10.2307/2528823.

See Also

paracou_6_abd, paracou_6_taxo

`paracou_6_taxo` *Taxonomy of Paracou plot 6 species*

Description

The taxonomy of species of the dataset paracou_6_abd. Distances in the tree are 1 (different species of the same genus), 2 (same family) or 3 (different families).

Usage

```
paracou_6_taxo
```

Format

An object of class ape::phylo, which is a phylogenetic tree.

Details

This dataset is from Paracou field station, French Guiana, managed by Cirad.

Source

Permanent data census of Paracou: `https://paracou.cirad.fr/`

See Also

paracou_6_abd, paracou_6_fundist

`phylo_divent` *Class phylo_divent*

Description

Methods for dendrograms of class "phylo_divent".

Usage

```
as_phylo_divent(tree)

is_phylo_divent(x)
```

Arguments

`tree`	an ultrametric, phylogenetic tree. May be an object of class phylo_divent, ape::phylo, ade4::phylog or stats::hclust.
`x`	An object of class "phylo_divent".

Details

`as_phylo_divent` calculates cuts and intervals of a phylogenetic tree and makes it available both in stats::hclust and ape::phylo formats. The conversion preprocesses the tree: it calculates cuts so that the tree can be reused efficiently by phylodiversity functions.

Value

`as_phylo_divent` returns a phylogenetic tree that is an object of class "phylo_divent".

Examples

```
# Paracou plot 6 species taxonomy
tree <- as_phylo_divent(mock_3sp_tree)
plot(tree)
```

`plot.phylo_divent` *Plot phylo_divent Objects*

Description

Plot objects of class "phylo_divent" produced by as_phylo_divent, that are phylogenetic trees.

Usage

```
## S3 method for class 'phylo_divent'
plot(x, ...)
```

Arguments

`x`	An object of class "phylo_divent".
`...`	Arguments passed to stats::plot.dendrogram.

Value

`NULL`. Called for side effects.

Examples

```
# Paracou plot 6 species taxonomy
tree <- as_phylo_divent(paracou_6_taxo)
plot(tree, leaflab = "none")
```

`plot.species_distribution` *Plot Profile Objects*

Description

Plot objects of class "species_distribution" produced by species_distribution and similar functions.

Usage

```
## S3 method for class 'species_distribution'
plot(
  x,
  type = c("RAC", "Metacommunity"),
  ...,
  fit_rac = FALSE,
  distribution = c("lnorm", "lseries", "geom", "bstick"),
  ylog = "y",
  main = NULL,
  xlab = "Rank",
  ylab = NULL,
  palette = "Set1"
)

## S3 method for class 'species_distribution'
autoplot(
  object,
  ...,
  fit_rac = FALSE,
  distribution = c("lnorm", "lseries", "geom", "bstick"),
  ylog = TRUE,
  main = NULL,
  xlab = "Rank",
  ylab = NULL,
  pch = ggplot2::GeomPoint$default_aes$shape,
  cex = ggplot2::GeomPoint$default_aes$size
)
```

Arguments

`x`	An object.
`type`	The type of plot. "RAC" (Rank-abundance curve, or Whittaker plot) or "Metacommunity" to represent species abundances of each community along with those of the metacommunity.
`...`	Additional arguments to be passed to plot. Unused elsewhere.
`fit_rac`	If `TRUE`, estimate a theoretical distribution and fit the data with it. RAC plot only.
`distribution`	The distribution of species abundances. May be "lnorm" (log-normal), "lseries" (log-series), "geom" (geometric) or "bstick" (broken stick). RAC plot only.
`ylog`	If `TRUE`, the Y-axis is in log-scale. RAC plot only.
`main`	The title of the plot.
`xlab`	The label of the X-axis. RAC plot only.
`ylab`	The label of the Y-axis.
`palette`	The name of a color palette, recognized by RColorBrewer::brewer.pal. RAC plot only.

`object`	An object of class species_distribution.
`pch`	The plotting characters. See graphics::points.
`cex`	The character expansion (size) of the points. See graphics::points.

Value

`NULL`. Called for side effects.

`probabilities` *Probabilities of Species*

Description

Estimate actual probabilities of species from a sample

Usage

```
probabilities(x, ...)

## S3 method for class 'numeric'
probabilities(
  x,
  estimator = c("naive", "Chao2013", "Chao2015", "ChaoShen"),
  unveiling = c("none", "uniform", "geometric"),
  richness_estimator = c("jackknife", "iChao1", "Chao1", "rarefy", "naive"),
  jack_alpha = 0.05,
  jack_max = 10,
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  q = 0,
  as_numeric = FALSE,
  ...,
  check_arguments = TRUE
)

## S3 method for class 'abundances'
probabilities(
  x,
  estimator = c("naive", "Chao2013", "Chao2015", "ChaoShen"),
  unveiling = c("none", "uniform", "geometric"),
  richness_estimator = c("jackknife", "iChao1", "Chao1", "rarefy", "naive"),
  jack_alpha = 0.05,
  jack_max = 10,
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  q = 0,
  ...,
  check_arguments = TRUE
)
```

Arguments

`x`	An object. It may be: • a numeric vector containing abundances. It may be named to track species names. • an object of class species_distribution.
`...`	Unused.
`estimator`	One of the estimators of a probability distribution: "naive" (the default value), or "Chao2013", "Chao2015", "ChaoShen" to estimate the probabilities of the observed species in the asymptotic distribution.
`unveiling`	One of the possible unveiling methods to estimate the probabilities of the unobserved species: "none" (default, no species is added), "uniform" (all unobserved species have the same probability) or "geometric" (the unobserved species distribution is geometric).
`richness_estimator`	An estimator of richness to evaluate the total number of species. "jackknife" is the default value. An alternative is "rarefy" to estimate the number of species such that the entropy of the asymptotic distribution rarefied to the observed sample size equals the actual entropy of the data.
`jack_alpha`	the risk level, 5% by default, used to optimize the jackknife order.
`jack_max`	the highest jackknife order allowed. Default is 10.
`coverage_estimator`	an estimator of sample coverage used by coverage.
`q`	The order of diversity. Default is 0 for richness. Used only to estimate asymptotic probability distributions when argument `richness_estimator` is "rarefy". Then, the number of unobserved species is fitted so that the entropy of order q of the asymptotic probability distribution at the observed sample size equals the actual entropy of the data.
`as_numeric`	if `TRUE`, a number or a numeric vector is returned rather than a tibble.
`check_arguments`	if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

`probabilities()` estimates a probability distribution from a sample. If the `estimator` is not "naive", the observed abundance distribution is used to estimate the actual probability distribution. The list of species is changed: zero-abundance species are cleared, and some unobserved species are added. First, observed species probabilities are estimated following Chao and Shen (2003), i.e. input probabilities are multiplied by the sample coverage, or according to more sophisticated models: Chao et al. (2013), a single-parameter model, or Chao and Jost (2015), a two-parameter model. The total probability of observed species equals the sample coverage. Then, the distribution of unobserved species can be unveiled: their number is estimated according to the `richness_estimator`. The coverage deficit (1 minus the sample coverage) is shared by the unobserved species equally (`unveiling` = "uniform", (Chao et al. 2013)) or according to a geometric distribution (`unveiling` = "geometric", (Chao and Jost 2015)).

Value

An object of class "probabilities", which is a species_distribution or a numeric vector with argument `as_numeric = TRUE`.

References

Chao A, Jost L (2015). "Estimating Diversity and Entropy Profiles via Discovery Rates of New Species." *Methods in Ecology and Evolution*, **6**(8), 873–882. doi:10.1111/2041210X.12349.

Chao A, Shen T (2003). "Nonparametric Estimation of Shannon's Index of Diversity When There Are Unseen Species in Sample." *Environmental and Ecological Statistics*, **10**(4), 429–443. doi:10.1023/A:1026096204727.

Chao A, Wang Y, Jost L (2013). "Entropy and the Species Accumulation Curve: A Novel Entropy Estimator via Discovery Rates of New Species." *Methods in Ecology and Evolution*, **4**(11), 1091–1100. doi:10.1111/2041210x.12108.

Examples

```
# Just transform abundances into probabilities
probabilities(paracou_6_abd)
# Estimate the distribution of probabilities from observed abundances (unveiled probabilities)
prob_unv <- probabilities(
  paracou_6_abd,
  estimator = "Chao2015",
  unveiling = "geometric",
  richness_estimator = "jackknife"
)
# Estimate entropy from the unveiled probabilities
ent_shannon(prob_unv)
# Identical to
ent_shannon(paracou_6_abd, estimator = "UnveilJ")
```

`profile_hill` *Diversity Profile of a Community*

Description

Calculate the diversity profile of a community, i.e. diversity (Hill numbers) against its order.

Usage

```
profile_hill(x, orders = seq(from = 0, to = 2, by = 0.1), ...)

## S3 method for class 'numeric'
profile_hill(
  x,
  orders = seq(from = 0, to = 2, by = 0.1),
 estimator = c("UnveilJ", "ChaoJost", "ChaoShen", "GenCov", "Grassberger", "Holste",
   "Marcon", "UnveilC", "UnveiliC", "ZhangGrabchak", "naive"),
  level = NULL,
  probability_estimator = c("Chao2015", "Chao2013", "ChaoShen", "naive"),
  unveiling = c("geometric", "uniform", "none"),
  richness_estimator = c("jackknife", "iChao1", "Chao1", "naive"),
  jack_alpha = 0.05,
  jack_max = 10,
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  sample_coverage = NULL,
  as_numeric = FALSE,
  n_simulations = 0,
  alpha = 0.05,
  bootstrap = c("Chao2015", "Marcon2012", "Chao2013"),
  show_progress = TRUE,
  ...,
  check_arguments = TRUE
)

## S3 method for class 'species_distribution'
profile_hill(
  x,
  orders = seq(from = 0, to = 2, by = 0.1),
 estimator = c("UnveilJ", "ChaoJost", "ChaoShen", "GenCov", "Grassberger", "Holste",
   "Marcon", "UnveilC", "UnveiliC", "ZhangGrabchak", "naive"),
  level = NULL,
  probability_estimator = c("Chao2015", "Chao2013", "ChaoShen", "naive"),
  unveiling = c("geometric", "uniform", "none"),
  richness_estimator = c("jackknife", "iChao1", "Chao1", "naive"),
  jack_alpha = 0.05,
  jack_max = 10,
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  gamma = FALSE,
  n_simulations = 0,
  alpha = 0.05,
  bootstrap = c("Chao2015", "Marcon2012", "Chao2013"),
  show_progress = TRUE,
  ...,
  check_arguments = TRUE
)
```

Arguments

`x`	An object, that may be a numeric vector containing abundances or probabilities, or an object of class abundances or probabilities.
`orders`	The orders of diversity used to build the profile.
`...`	Unused.

`estimator`	An estimator of entropy.
`level`	the level of interpolation or extrapolation. It may be a sample size (an integer) or a sample coverage (a number between 0 and 1). If not `NULL`, the asymptotic `estimator` is ignored.
`probability_estimator`	a string containing one of the possible estimators of the probability distribution (see probabilities). Used only for extrapolation.
`unveiling`	a string containing one of the possible unveiling methods to estimate the probabilities of the unobserved species (see probabilities). Used only for extrapolation.
`richness_estimator`	an estimator of richness to evaluate the total number of species, see div_richness. used for interpolation and extrapolation.
`jack_alpha`	the risk level, 5% by default, used to optimize the jackknife order.
`jack_max`	the highest jackknife order allowed. Default is 10.
`coverage_estimator`	an estimator of sample coverage used by coverage.
`sample_coverage`	the sample coverage of `x` calculated elsewhere. Used to calculate the gamma diversity of meta-communities, see details.
`as_numeric`	if `TRUE`, a number or a numeric vector is returned rather than a tibble.
`n_simulations`	The number of simulations used to estimate the confidence envelope of the profile.
`alpha`	The risk level, 5% by default, of the confidence envelope of the profile.
`bootstrap`	The method used to obtain the probabilities to generate bootstrapped communities from observed abundances. If "Marcon2012", the probabilities are simply the abundances divided by the total number of individuals (Marcon et al. 2012). If "Chao2013" or "Chao2015" (by default), a more sophisticated approach is used (see as_probabilities) following Chao et al. (2013) or Chao and Jost (2015).
`show_progress`	if TRUE, a progress bar is shown during long computations.
`check_arguments`	if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.
`gamma`	if `TRUE`, γ diversity, i.e. diversity of the metacommunity, is computed.

Details

A bootstrap confidence interval can be produced by simulating communities (their number is `n_simulations`) with rcommunity and calculating their profiles. Simulating communities implies a downward bias in the estimation: rare species of the actual community may have abundance zero in simulated communities. Simulated diversity values are recentered so that their mean is that of the actual community.

Value

A tibble with the site names, the estimators used and the estimated diversity at each order. This is an object of class "profile" that can be plotted.

References

Chao A, Jost L (2015). "Estimating Diversity and Entropy Profiles via Discovery Rates of New Species." *Methods in Ecology and Evolution*, **6**(8), 873–882. doi:10.1111/2041210X.12349.

Chao A, Wang Y, Jost L (2013). "Entropy and the Species Accumulation Curve: A Novel Entropy Estimator via Discovery Rates of New Species." *Methods in Ecology and Evolution*, **4**(11), 1091–1100. doi:10.1111/2041210x.12108.

Marcon E, Hérault B, Baraloto C, Lang G (2012). "The Decomposition of Shannon's Entropy and a Confidence Interval for *Beta* Diversity." *Oikos*, **121**(4), 516–522. doi:10.1111/j.16000706.2011.19267.x.

Examples

```
autoplot(profile_hill(paracou_6_abd))
```

`profile_phylo` *Phylogenetic Diversity Profile of a Community*

Description

Calculate the diversity profile of a community, i.e. its phylogenetic diversity against its order.

Usage

```
profile_phylo(x, tree, orders = seq(from = 0, to = 2, by = 0.1), ...)

## S3 method for class 'numeric'
profile_phylo(
  x,
  tree,
  orders = seq(from = 0, to = 2, by = 0.1),
  normalize = TRUE,
 estimator = c("UnveilJ", "ChaoJost", "ChaoShen", "GenCov", "Grassberger", "Holste",
    "Marcon", "UnveilC", "UnveiliC", "ZhangGrabchak", "naive"),
  level = NULL,
  probability_estimator = c("Chao2015", "Chao2013", "ChaoShen", "naive"),
  unveiling = c("geometric", "uniform", "none"),
  richness_estimator = c("jackknife", "iChao1", "Chao1", "naive"),
  jack_alpha = 0.05,
  jack_max = 10,
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  sample_coverage = NULL,
  as_numeric = FALSE,
  n_simulations = 0,
  alpha = 0.05,
  bootstrap = c("Chao2015", "Marcon2012", "Chao2013"),
  show_progress = TRUE,
  ...,
  check_arguments = TRUE
)

## S3 method for class 'species_distribution'
profile_phylo(
  x,
  tree,
  orders = seq(from = 0, to = 2, by = 0.1),
  normalize = TRUE,
 estimator = c("UnveilJ", "ChaoJost", "ChaoShen", "GenCov", "Grassberger", "Holste",
    "Marcon", "UnveilC", "UnveiliC", "ZhangGrabchak", "naive"),
  level = NULL,
  probability_estimator = c("Chao2015", "Chao2013", "ChaoShen", "naive"),
  unveiling = c("geometric", "uniform", "none"),
  richness_estimator = c("jackknife", "iChao1", "Chao1", "naive"),
  jack_alpha = 0.05,
  jack_max = 10,
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  gamma = FALSE,
  n_simulations = 0,
  alpha = 0.05,
  bootstrap = c("Chao2015", "Marcon2012", "Chao2013"),
  show_progress = TRUE,
  ...,
  check_arguments = TRUE
)
```

Arguments

`x`	An object, that may be a numeric vector containing abundances or probabilities, or an object of class abundances or probabilities.
`tree`	an ultrametric, phylogenetic tree. May be an object of class phylo_divent, ape::phylo, ade4::phylog or stats::hclust.
`orders`	The orders of diversity used to build the profile.
`...`	Unused.
`normalize`	if `TRUE`, phylogenetic is normalized: the height of the tree is set to 1.
`estimator`	An estimator of entropy.
`level`	the level of interpolation or extrapolation. It may be a sample size (an integer) or a sample coverage (a number between 0 and 1). If not `NULL`, the asymptotic `estimator` is ignored.
`probability_estimator`	a string containing one of the possible estimators of the probability distribution (see probabilities). Used only for extrapolation.

`unveiling`	a string containing one of the possible unveiling methods to estimate the probabilities of the unobserved species (see probabilities). Used only for extrapolation.
`richness_estimator`	an estimator of richness to evaluate the total number of species, see div_richness. used for interpolation and extrapolation.
`jack_alpha`	the risk level, 5% by default, used to optimize the jackknife order.
`jack_max`	the highest jackknife order allowed. Default is 10.
`coverage_estimator`	an estimator of sample coverage used by coverage.
`sample_coverage`	the sample coverage of `x` calculated elsewhere. Used to calculate the gamma diversity of meta-communities, see details.
`as_numeric`	if `TRUE`, a number or a numeric vector is returned rather than a tibble.
`n_simulations`	The number of simulations used to estimate the confidence envelope of the profile.
`alpha`	The risk level, 5% by default, of the confidence envelope of the profile.
`bootstrap`	The method used to obtain the probabilities to generate bootstrapped communities from observed abundances. If "Marcon2012", the probabilities are simply the abundances divided by the total number of individuals (Marcon et al. 2012). If "Chao2013" or "Chao2015" (by default), a more sophisticated approach is used (see as_probabilities) following Chao et al. (2013) or Chao and Jost (2015).
`show_progress`	if TRUE, a progress bar is shown during long computations.
`check_arguments`	if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.
`gamma`	if `TRUE`, γ diversity, i.e. diversity of the metacommunity, is computed.

Details

A bootstrap confidence interval can be produced by simulating communities (their number is `n_simulations`) with rcommunity and calculating their profiles. Simulating communities implies a downward bias in the estimation: rare species of the actual community may have abundance zero in simulated communities. Simulated diversity values are recentered so that their mean is that of the actual community.

Value

A tibble with the site names, the estimators used and the estimated diversity at each order. This is an object of class "profile" that can be plotted.

References

Chao A, Jost L (2015). “Estimating Diversity and Entropy Profiles via Discovery Rates of New Species.” *Methods in Ecology and Evolution*, **6**(8), 873–882. doi:10.1111/2041210X.12349.

Chao A, Wang Y, Jost L (2013). “Entropy and the Species Accumulation Curve: A Novel Entropy Estimator via Discovery Rates of New Species.” *Methods in Ecology and Evolution*, **4**(11), 1091–1100. doi:10.1111/2041210x.12108.

Marcon E, Hérault B, Baraloto C, Lang G (2012). “The Decomposition of Shannon’s Entropy and a Confidence Interval for *Beta* Diversity.” *Oikos*, **121**(4), 516–522. doi:10.1111/j.16000706.2011.19267.x.

Examples

```
profile_phylo(paracou_6_abd, tree = paracou_6_taxo)
```

`profile_similarity` *Similarity-Based Diversity Profile of a Community*

Description

Calculate the diversity profile of a community, i.e. its similarity-based diversity against its order.

Usage

```
profile_similarity(
  x,
  similarities,
  orders = seq(from = 0, to = 2, by = 0.1),
  ...
)

## S3 method for class 'numeric'
profile_similarity(
  x,
  similarities = diag(length(x)),
  orders = seq(from = 0, to = 2, by = 0.1),
 estimator = c("UnveilJ", "Max", "ChaoShen", "MarconZhang", "UnveilC", "UnveiliC",
   "naive"),
  probability_estimator = c("Chao2015", "Chao2013", "ChaoShen", "naive"),
  unveiling = c("geometric", "uniform", "none"),
  richness_estimator = c("jackknife", "iChao1", "Chao1", "naive"),
  jack_alpha = 0.05,
  jack_max = 10,
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  sample_coverage = NULL,
  as_numeric = FALSE,
  n_simulations = 0,
  alpha = 0.05,
  bootstrap = c("Chao2015", "Marcon2012", "Chao2013"),
  show_progress = TRUE,
  ...,
  check_arguments = TRUE
)

## S3 method for class 'species_distribution'
profile_similarity(
  x,
  similarities = diag(sum(!colnames(x) %in% non_species_columns)),
  orders = seq(from = 0, to = 2, by = 0.1),
 estimator = c("UnveilJ", "Max", "ChaoShen", "MarconZhang", "UnveilC", "UnveiliC",
    "naive"),
  probability_estimator = c("Chao2015", "Chao2013", "ChaoShen", "naive"),
  unveiling = c("geometric", "uniform", "none"),
  jack_alpha = 0.05,
  jack_max = 10,
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  gamma = FALSE,
  n_simulations = 0,
  alpha = 0.05,
  bootstrap = c("Chao2015", "Marcon2012", "Chao2013"),
  show_progress = TRUE,
  ...,
  check_arguments = TRUE
)
```

Arguments

`x`	An object, that may be a numeric vector containing abundances or probabilities, or an object of class abundances or probabilities.
`similarities`	a similarity matrix, that can be obtained by fun_similarity. Its default value is the identity matrix.
`orders`	The orders of diversity used to build the profile.
`...`	Unused.
`estimator`	An estimator of entropy.
`probability_estimator`	a string containing one of the possible estimators of the probability distribution (see probabilities). Used only for extrapolation.
`unveiling`	a string containing one of the possible unveiling methods to estimate the probabilities of the unobserved species (see probabilities). Used only for extrapolation.
`richness_estimator`	an estimator of richness to evaluate the total number of species, see div_richness. used for interpolation and extrapolation.
`jack_alpha`	the risk level, 5% by default, used to optimize the jackknife order.
`jack_max`	the highest jackknife order allowed. Default is 10.
`coverage_estimator`	an estimator of sample coverage used by coverage.

`sample_coverage`	the sample coverage of `x` calculated elsewhere. Used to calculate the gamma diversity of meta-communities, see details.
`as_numeric`	if `TRUE`, a number or a numeric vector is returned rather than a tibble.
`n_simulations`	The number of simulations used to estimate the confidence envelope of the profile.
`alpha`	The risk level, 5% by default, of the confidence envelope of the profile.
`bootstrap`	The method used to obtain the probabilities to generate bootstrapped communities from observed abundances. If "Marcon2012", the probabilities are simply the abundances divided by the total number of individuals (Marcon et al. 2012). If "Chao2013" or "Chao2015" (by default), a more sophisticated approach is used (see as_probabilities) following Chao et al. (2013) or Chao and Jost (2015).
`show_progress`	if TRUE, a progress bar is shown during long computations.
`check_arguments`	if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.
`gamma`	if `TRUE`, γ diversity, i.e. diversity of the metacommunity, is computed.

Details

A bootstrap confidence interval can be produced by simulating communities (their number is `n_simulations`) with rcommunity and calculating their profiles. Simulating communities implies a downward bias in the estimation: rare species of the actual community may have abundance zero in simulated communities. Simulated diversity values are recentered so that their mean is that of the actual community.

Value

A tibble with the site names, the estimators used and the estimated diversity at each order. This is an object of class "profile" that can be plotted.

References

Chao A, Jost L (2015). “Estimating Diversity and Entropy Profiles via Discovery Rates of New Species.” *Methods in Ecology and Evolution*, **6**(8), 873–882. doi:10.1111/2041210X.12349.

Chao A, Wang Y, Jost L (2013). “Entropy and the Species Accumulation Curve: A Novel Entropy Estimator via Discovery Rates of New Species.” *Methods in Ecology and Evolution*, **4**(11), 1091–1100. doi:10.1111/2041210x.12108.

Marcon E, Hérault B, Baraloto C, Lang G (2012). “The Decomposition of Shannon’s Entropy and a Confidence Interval for *Beta* Diversity.” *Oikos*, **121**(4), 516–522. doi:10.1111/j.16000706.2011.19267.x.

Examples

```
# Similarity matrix
Z <- fun_similarity(paracou_6_fundist)
# Profile
```

```
profile_similarity(paracou_6_abd, similarities = Z, q = 2)
```

`rcommunity` *Random communities*

Description

`rcommunity()` draws random communities according to a probability distribution. `rspcommunity()` extends it by spatializing the random communities.

Usage

```
rcommunity(
  n,
  size = sum(abd),
  prob = NULL,
  abd = NULL,
  bootstrap = c("Chao2015", "Marcon2012", "Chao2013"),
  species_number = 300,
  distribution = c("lnorm", "lseries", "geom", "bstick"),
  sd_lnorm = 1,
  prob_geom = 0.1,
  fisher_alpha = 40,
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  check_arguments = TRUE
)

rspcommunity(
  n,
  size = sum(abd),
  prob = NULL,
  abd = NULL,
  bootstrap = c("Chao2015", "Marcon2012", "Chao2013"),
  species_number = 300,
  distribution = c("lnorm", "lseries", "geom", "bstick"),
  sd_lnorm = 1,
  prob_geom = 0.1,
  fisher_alpha = 40,
  coverage_estimator = c("ZhangHuang", "Chao", "Turing", "Good"),
  spatial = c("Binomial", "Thomas"),
  thomas_scale = 0.2,
  thomas_mu = 10,
  win = spatstat.geom::owin(),
  species_names = NULL,
  weight_distribution = c("Uniform", "Weibull", "Exponential"),
  w_min = 1,
  w_max = 1,
  w_mean = 20,
  weibull_scale = 20,
  weibull_shape = 2,
  check_arguments = TRUE
)
```

Arguments

`n`	the number of communities to draw.
`size`	The number of individuals to draw in each community.
`prob`	a numeric vector containing probabilities.
`abd`	a numeric vector containing abundances.
`bootstrap`	The method used to obtain the probabilities to generate bootstrapped communities from observed abundances. If "Marcon2012", the probabilities are simply the abundances divided by the total number of individuals (Marcon et al. 2012). If "Chao2013" or "Chao2015" (by default), a more sophisticated approach is used (see as_probabilities) following Chao et al. (2013) or Chao and Jost (2015).
`species_number`	The number of species.
`distribution`	The distribution of species abundances. May be "lnorm" (log-normal), "lseries" (log-series), "geom" (geometric) or "bstick" (broken stick).
`sd_lnorm`	the simulated log-normal distribution standard deviation. This is the standard deviation on the log scale.
`prob_geom`	the proportion of resources taken by successive species of the geometric distribution.
`fisher_alpha`	Fisher's α in the log-series distribution.
`coverage_estimator`	an estimator of sample coverage used by coverage.
`check_arguments`	if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.
`spatial`	the spatial distribution of points. May be "Binomial" (a completely random point pattern except for its fixed number of points) or "Thomas" for a clustered point pattern with parameters `scale` and `mu`.
`thomas_scale`	in Thomas point patterns, the standard deviation of random displacement (along each coordinate axis) of a point from its cluster center.
`thomas_mu`	in Thomas point patterns, the mean number of points per cluster. The intensity of the Poisson process of cluster centers is calculated as the number of points (`size`) per area divided by `thomas_mu`.
`win`	the window containing the point pattern. It is an spatstat.geom::owin object. Default is a 1x1 square.
`species_names`	a vector of characters or of factors containing the possible species names.

`weight_distribution`	the distribution of point weights. By default, all weight are 1. May be "uniform" for a uniform distribution between `w_min` and `w_max`, "weibull" with parameters `w_min`, `weibull_scale` and `shape` or "exponential" with parameter `w_mean`.
`w_min`	the minimum weight in a uniform or Weibull distribution.
`w_max`	the maximum weight in a uniform distribution.
`w_mean`	the mean weight in an exponential distribution (i.e. the negative of the inverse of the decay rate).
`weibull_scale`	the scale parameter in a Weibull distribution.
`weibull_shape`	the shape parameter in a Weibull distribution.

Details

Communities of fixed `size` are drawn in a multinomial distribution according to the distribution of probabilities provided by `prob`. An abundance vector `abd` may be used instead of probabilities, then `size` is by default the total number of individuals in the vector. Random communities can be built by drawing in a multinomial law following Marcon et al. (2012), or trying to estimate the distribution of the actual community with as_probabilities. If `bootstrap` is "Chao2013", the distribution is estimated by a single parameter model and unobserved species are given equal probabilities. If `bootstrap` is "Chao2015", a two-parameter model is used and unobserved species follow a geometric distribution.

Alternatively, the probabilities may be drawn following a classical distribution: either lognormal ("lnorm") (Preston 1948) with given standard deviation (`sd_lnorm`; note that the mean is actually a normalizing constant. Its value is set equal to 0 for the simulation of the normal distribution of unnormalized log-abundances), log-series ("lseries") (Fisher et al. 1943) with parameter `fisher_alpha`, geometric ("geom") (Motomura 1932) with parameter `prob_geom`, or broken stick ("bstick") (MacArthur 1957). The number of simulated species is fixed by `species_number`, except for "lseries" where it is obtained from `fisher_alpha` and `size`: $S = \alpha \ln(1 + size/\alpha)$. Note that the probabilities are drawn once only. If the number of communities to draw, `n`, is greater than 1, then they are drawn in a multinomial distribution following these probabilities.

Log-normal, log-series and broken-stick distributions are stochastic. The geometric distribution is completely determined by its parameters.

Spatialized communities include the location of individuals in a window, in a a dbmss::wmppp object. Several point processes are available, namely binomial (points are uniformly distributed in the window) and Thomas (1949), which is clustered.

Point weights, that may be for instance the size of the trees in a forest community, can be uniform, follow a Weibull or a negative exponential distribution. The latter describe well the diameter distribution of trees in a forest (Rennolls et al. 1985; Turner 2004).

Value

`rcommunity()` returns an object of class abundances.

`rspcommunity()` returns either a spatialized community, which is a dbmss::wmppp object , with `PointType` values as species names if `n=1` or an object of class ppplist (see spatstat.geom::solist) if `n>1`.

References

Chao A, Jost L (2015). “Estimating Diversity and Entropy Profiles via Discovery Rates of New Species.” *Methods in Ecology and Evolution*, **6**(8), 873–882. doi:10.1111/2041210X.12349.

Chao A, Wang Y, Jost L (2013). “Entropy and the Species Accumulation Curve: A Novel Entropy Estimator via Discovery Rates of New Species.” *Methods in Ecology and Evolution*, **4**(11), 1091–1100. doi:10.1111/2041210x.12108.

Fisher RA, Corbet AS, Williams CB (1943). “The Relation between the Number of Species and the Number of Individuals in a Random Sample of an Animal Population.” *Journal of Animal Ecology*, **12**, 42–58. doi:10.2307/1411.

MacArthur RH (1957). “On the Relative Abundance of Bird Species.” *Proceedings of the National Academy of Sciences of the United States of America*, **43**(3), 293–295. doi:10.1073/pnas.43.3.293, 89566.

Marcon E, Hérault B, Baraloto C, Lang G (2012). “The Decomposition of Shannon’s Entropy and a Confidence Interval for *Beta* Diversity.” *Oikos*, **121**(4), 516–522. doi:10.1111/j.16000706.2011.19267.x.

Motomura I (1932). “On the statistical treatment of communities.” *Zoological Magazine*, **44**, 379–383.

Preston FW (1948). “The Commonness, and Rarity, of Species.” *Ecology*, **29**(3), 254–283. doi:10.2307/1930989.

Rennolls K, Geary DN, Rollinson TJD (1985). “Characterizing Diameter Distributions by the Use of the Weibull Distribution.” *Forestry*, **58**(1), 57–66. ISSN 0015-752X, 1464-3626, doi:10.1093/forestry/58.1.57, `2024-11-07`.

Thomas M (1949). “A Generalization of Poisson’s Binomial Limit for Use in Ecology.” *Biometrika*, **36**(1/2), 18–25. doi:10.2307/2332526, 2332526.

Turner IM (2004). *The Ecology of Trees in the Tropical Rain Forest*, 2nd edition. Cambridge University Press. ISBN 978-0-521-80183-6, doi:10.1017/CBO9780511542206, `2024-11-07`.
Chao A, Jost L (2015). “Estimating Diversity and Entropy Profiles via Discovery Rates of New Species.” *Methods in Ecology and Evolution*, **6**(8), 873–882. doi:10.1111/2041210X.12349.

Chao A, Wang Y, Jost L (2013). “Entropy and the Species Accumulation Curve: A Novel Entropy Estimator via Discovery Rates of New Species.” *Methods in Ecology and Evolution*, **4**(11), 1091–1100. doi:10.1111/2041210x.12108.

Fisher RA, Corbet AS, Williams CB (1943). “The Relation between the Number of Species and the Number of Individuals in a Random Sample of an Animal Population.” *Journal of Animal Ecology*, **12**, 42–58. doi:10.2307/1411.

MacArthur RH (1957). “On the Relative Abundance of Bird Species.” *Proceedings of the National Academy of Sciences of the United States of America*, **43**(3), 293–295. doi:10.1073/pnas.43.3.293, 89566.

Marcon E, Hérault B, Baraloto C, Lang G (2012). "The Decomposition of Shannon's Entropy and a Confidence Interval for *Beta* Diversity." *Oikos*, **121**(4), 516–522. doi:10.1111/j.16000706.2011.19267.x.

Motomura I (1932). "On the statistical treatment of communities." *Zoological Magazine*, **44**, 379–383.

Preston FW (1948). "The Commonness, and Rarity, of Species." *Ecology*, **29**(3), 254–283. doi:10.2307/1930989.

Rennolls K, Geary DN, Rollinson TJD (1985). "Characterizing Diameter Distributions by the Use of the Weibull Distribution." *Forestry*, **58**(1), 57–66. ISSN 0015-752X, 1464-3626, doi:10.1093/forestry/58.1.57, 2024-11-07.

Thomas M (1949). "A Generalization of Poisson's Binomial Limit for Use in Ecology." *Biometrika*, **36**(1/2), 18–25. doi:10.2307/2332526, 2332526.

Turner IM (2004). *The Ecology of Trees in the Tropical Rain Forest*, 2nd edition. Cambridge University Press. ISBN 978-0-521-80183-6, doi:10.1017/CBO9780511542206, 2024-11-07.

Examples

```
# Generate a community made of 100000 individuals among 300 species and fit it
abundances <- rcommunity(n = 1, size = 1E5,
  species_number = 300, distribution = "lnorm")
autoplot(abundances)
X <- rspcommunity(1, size = 30, species_number = 5)
autoplot(X)
```

---

rlseries *Log-Series Distribution*

Description

Random generation for the log-series distribution.

Usage

```
rlseries(n, size, fisher_alpha, show_progress = TRUE, check_arguments = TRUE)
```

Arguments

`n`	the number of observations.
`size`	The size of the distribution.
`fisher_alpha`	Fisher's α in the log-series distribution.
`show_progress`	if TRUE, a progress bar is shown during long computations.

`check_arguments`
: if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

Fast implementation of the random generation of a log-series distribution (Fisher et al. 1943).

The complete set of functions (including density, distribution function and quantiles) can be found in package *sads* but this implementation of the random generation is much faster.

If `size` is too large, i.e. `size` + 1 can't be distinguished from `size` due to rounding, then an error is raised.

Value

A numeric vector with the random values drawn from the log-series distribution.

References

Fisher RA, Corbet AS, Williams CB (1943). "The Relation between the Number of Species and the Number of Individuals in a Random Sample of an Animal Population." *Journal of Animal Ecology*, **12**, 42–58. doi:10.2307/1411.

Examples

```
# Generate a community made of 10000 individuals with alpha=40
size <- 1E4
fisher_alpha <- 40
species_number <- fisher_alpha * log(1 + size / fisher_alpha)
abundances <- rlseries(species_number, size = 1E5, fisher_alpha = 40)
# rcommunity() may be a better choice here
autoplot(rcommunity(1, size = 1E4, fisher_alpha = 40, distribution = "lseries"))
```

`species_distribution` *Species Distributions*

Description

A Species Distribution is a tibble::tibble containing species abundances or probabilities. Rows of the tibble are communities and column are species. Values are either abundances or probabilities. Special columns contain the site names, and their weights (e.g. their area or number of individuals): their names must be "site" and "weight". All other column names are considered as species names.

Usage

```
species_distribution(x, names = NULL, weights = NULL, check_arguments = TRUE)

as_species_distribution(x, ...)

## S3 method for class 'numeric'
as_species_distribution(x, ..., check_arguments = TRUE)

## S3 method for class 'matrix'
as_species_distribution(
  x,
  names = NULL,
  weights = NULL,
  ...,
  check_arguments = TRUE
)

## S3 method for class 'data.frame'
as_species_distribution(x, ..., check_arguments = TRUE)

## S3 method for class 'wmppp'
as_species_distribution(x, ..., check_arguments = TRUE)

## S3 method for class 'character'
as_species_distribution(x, ..., check_arguments = TRUE)

## S3 method for class 'factor'
as_species_distribution(x, ..., check_arguments = TRUE)

is_species_distribution(x)

as_probabilities(x, ...)

## S3 method for class 'numeric'
as_probabilities(x, ..., check_arguments = TRUE)

## S3 method for class 'matrix'
as_probabilities(x, names = NULL, weights = NULL, ..., check_arguments = TRUE)

## S3 method for class 'data.frame'
as_probabilities(x, ..., check_arguments = TRUE)

## S3 method for class 'wmppp'
as_probabilities(x, ..., check_arguments = TRUE)

## S3 method for class 'character'
as_probabilities(x, ..., check_arguments = TRUE)
```

```
## S3 method for class 'factor'
as_probabilities(x, ..., check_arguments = TRUE)

is_probabilities(x)

abundances(
  x,
  round = TRUE,
  names = NULL,
  weights = NULL,
  check_arguments = TRUE
)

as_abundances(x, ...)

## S3 method for class 'numeric'
as_abundances(x, round = TRUE, ..., check_arguments = TRUE)

## S3 method for class 'matrix'
as_abundances(
  x,
  round = TRUE,
  names = NULL,
  weights = NULL,
  ...,
  check_arguments = TRUE
)

## S3 method for class 'data.frame'
as_abundances(x, ..., check_arguments = TRUE)

## S3 method for class 'wmppp'
as_abundances(x, ..., check_arguments = TRUE)

## S3 method for class 'character'
as_abundances(x, ..., check_arguments = TRUE)

## S3 method for class 'factor'
as_abundances(x, ..., check_arguments = TRUE)

is_abundances(x)

## S3 method for class 'species_distribution'
as.matrix(x, use.names = TRUE, ...)

## S3 method for class 'species_distribution'
as.double(x, use.names = TRUE, ...)
```

```
## S3 method for class 'species_distribution'
as.numeric(x, use.names = TRUE, ...)
```

Arguments

`x`	an object.
`names`	The names of the species distributions.
`weights`	the weights of the sites of the species distributions.
`check_arguments`	if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.
`...`	Unused.
`round`	If `TRUE`, the values of `x` are converted to integers.
`use.names`	If `TRUE`, the names of the `species_distribution` are kept in the matrix or vector they are converted to.

Details

`species_distribution` objects include `abundances` and `probabilities` objects.

`species_distribution()` creates a `species_distribution` object from a vector or a matrix or a dataframe.

`as_species_distribution()`, `as_abundances()` and `as_probabilities` format the numeric, matrix or dataframe `x` so that appropriate versions of community functions (generic methods such as plot or div_richness) are applied. Abundance values are rounded (by default) to the nearest integer. They also accept a dbmss::wmppp objects, i.e. a weighted, marked planar point pattern and count the abundances of point types, character and factor objects.

`as_probabilities()` normalizes the vector `x` so that it sums to 1. It gives the same output as `probabilities()` with `estimator = "naive"`.

`species_distribution` objects objects can be plotted by plot and autoplot.

Value

An object of classes `species_distribution` and `abundances` or `probabilities`.

`as.double()` and its synonymous `as.numeric()` return a numeric vector that contains species abundances or probabilities of a single-row `species_distribution`. `as.matrix()` returns a numeric matrix if the `species_distribution` contains several rows. These are methods of the generic functions for class `species_distribution`.

Examples

```
# Paracou data is a tibble
paracou_6_abd
# Class
class(paracou_6_abd)
is_species_distribution(paracou_6_abd)
# Whittaker plot fitted by a log-normal distribution
autoplot(paracou_6_abd[1,], fit_rac = TRUE, distribution = "lnorm")
```

```
# Character vectors
as_abundances(c("A", "C", "B", "C"))
```

Index

∗ **datasets**
mock_3sp, 63
paracou_6, 64
paracou_6_fundist, 65
paracou_6_taxo, 65

abd_freq_count, 4
abd_species, 5
abd_sum *(*abd_species*)*, 5
abundances, *5*, *8*, *11*, *25*, *29, 30*, *32*, *34*, *37*, *39*, *41*, *43*, *45*, *48*, *50*, *54*, *58*, *62–64*, *72*, *75*, *78*, *82*
abundances *(*species_distribution*)*, 85
accum_div_phylo, 6
accum_ent_phylo *(*accum_div_phylo*)*, 6
accum_hill, 9, *19*, *27*
accum_mixing *(*accum_sp_hill*)*, 15
accum_sp, 12, *17*
accum_sp_diversity, *17*
accum_sp_diversity *(*accum_sp*)*, 12
accum_sp_entropy, *17*
accum_sp_entropy *(*accum_sp*)*, 12
accum_sp_hill, 15
accum_sp_mixing, *17*
accum_sp_mixing *(*accum_sp*)*, 12
accum_sp_tsallis *(*accum_sp_hill*)*, 15
accum_tsallis, *33*, *41*, *46*, *50*, *54*
accum_tsallis *(*accum_hill*)*, 9
ade4::phylog, *8*, *32*, *41*, *43*, *66*, *75*
alphahull, 18
alphahull::ashape(), *18*
ape::phylo, *8*, *32*, *41*, *43*, *63*, *66*, *75*
as.double.species_distribution *(*species_distribution*)*, 85
as.matrix.species_distribution *(*species_distribution*)*, 85
as.numeric.species_distribution *(*species_distribution*)*, 85
as_abundances *(*species_distribution*)*, 85
as_phylo_divent, *67*
as_phylo_divent *(*phylo_divent*)*, 66
as_probabilities, *73*, *76*, *79*, *81, 82*
as_probabilities *(*species_distribution*)*, 85
as_species_distribution *(*species_distribution*)*, 85
autoplot, *88*
autoplot.accum_sp *(*accum_sp*)*, 12
autoplot.accumulation, 19
autoplot.profile, 20
autoplot.species_distribution *(*plot.species_distribution*)*, 67
autoplot.wmppp, 21

beta, *56*

coverage, *4*, *8*, *11*, 22, *26*, *30*, *33*, *35*, *37*, *41*, *45*, *48*, *50*, *54*, *70*, *73*, *76*, *78*, *81*
coverage_to_size *(*coverage*)*, 22

dbmss::autoplot.wmppp, *21*
dbmss::wmppp, *15, 16*, *52*, *64*, *82*, *88*
div_hill, 24, *33*, *41*, *46*, *50*, *54*
div_hurlbert, 28, *39*
div_part, 30
div_phylo, 31
div_richness, *4*, *8*, *11*, *17*, *26*, *30*, *33*, 34, *41*, *45*, *50*, *54*, *73*, *76*, *78*, *88*
div_similarity, 36
divent *(*divent-package*)*, 3
divent-package, 3

e_n_q, 56
ent_hurlbert, 38
ent_phylo, 40, *43*
ent_rao, 42
ent_shannon, 44
ent_similarity, *43*, 47
ent_simpson, 49
ent_sp_simpson, 51

ent_sp_simpsonEnvelope
(ent_sp_simpson), 51
ent_tsallis, *8*, *12*, 53
exp_q, 55

fit_rac, 57
fun_ordinariness, 58
fun_similarity, *37*, *48*, *59*, 60, *78*

ggplot2::ggplot, *15*, *19*, *20*, *22*
ggplot2::scale_colour_brewer, *22*
graphics::points, *69*

is_abundances (species_distribution), 85
is_phylo_divent (phylo_divent), 66
is_probabilities
(species_distribution), 85
is_species_distribution
(species_distribution), 85

ln_q, *55*, 61

metacommunity, *27*, 62
mock_3sp, 63
mock_3sp_abd (mock_3sp), 63
mock_3sp_dist (mock_3sp), 63
mock_3sp_tree (mock_3sp), 63

paracou_6, 64
paracou_6_abd, *65*, *66*
paracou_6_abd (paracou_6), 64
paracou_6_fundist, *64*, 65, *66*
paracou_6_taxo, *64*, *65*, 65
paracou_6_wmppp (paracou_6), 64
phylo_divent, *8*, *32*, *41*, *43*, *66*, 66, *75*
plot, *14*, *68*, *88*
plot.accum_sp (accum_sp), 12
plot.phylo_divent, 67
plot.species_distribution, 67
plot_map (accum_sp), 12
prob_species (abd_species), 5
probabilities, *4*, *8*, *11*, *25*, *26*, *29*, *30*, *32*, *34*, *35*, *37*, *39*, *41*, *43*, *45*, *48*, *50*, *54*, 69, *72*, *73*, *75*, *76*, *78*
profile_hill, *20*, 71
profile_phylo, 74
profile_similarity, 77

RColorBrewer::brewer.pal, *68*
rcommunity, *73*, *76*, *79*, 80
rlseries, 84
rspcommunity (rcommunity), 80

spatstat.explore::bw.smoothppp, *14*
spatstat.explore::density.ppp, *14*
spatstat.explore::envelope, *52*
spatstat.explore::fv.object, *52*
spatstat.explore::Kest, *52*
spatstat.geom::convexhull, *18*
spatstat.geom::im, *15*
spatstat.geom::owin, *81*
spatstat.geom::owin.object, *18*
spatstat.geom::plot.im, *14*
spatstat.geom::ppp.object, *18*
spatstat.geom::solist, *82*
species_distribution, *5*, *59*, *67*, *69–71*, 85
stats::dist, *43*, *60*
stats::hclust, *8*, *32*, *41*, *43*, *66*, *75*
stats::plot.dendrogram, *67*

tibble::tibble, *64*, *85*