Package ‘entropart’

February 7, 2025

Type Package

Title Entropy Partitioning to Measure Diversity

Version 1.6-16

Description Measurement and partitioning of diversity, based on Tsallis entropy, following Marcon and Herault (2015) <doi:10.18637/jss.v067.i08>.
'entropart' provides functions to calculate alpha, beta and gamma diversity of communities, including phylogenetic and functional diversity.
Estimation-bias corrections are available.

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

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

License GNU General Public License

Imports ape, EntropyEstimation, ggplot2, ggpubr, graphics, grDevices, parallel, reshape2, rlang, stats, tibble, utils, vegan

Suggests ade4, knitr, pkgdown, rmarkdown, testthat

LazyData true

VignetteBuilder knitr

SystemRequirements pandoc

Encoding UTF-8

NeedsCompilation no

Author Eric Marcon [aut, cre] (<`https://orcid.org/0000-0002-5249-321X`>),
Bruno Herault [aut] (<`https://orcid.org/0000-0002-6950-7286`>)

Maintainer Eric Marcon <`eric.marcon@agroparistech.fr`>

Repository CRAN

Date/Publication 2025-02-07 12:40:02 UTC

Contents

`entropart-package` *Entropy Partitioning to Measure Diversity*

Description

Functions to calculate alpha, beta and gamma diversity of communities, including phylogenetic and functional diversity.

Estimation-bias corrections are available.

Details

In the entropart package, individuals of different "species" are counted in several "communities" which may (or not) be agregated to define a "metacommunity". In the metacommunity, the probability to find a species in the weighted average of probabilities in communities. This is a naming convention, which may correspond to plots in a forest inventory or any data organized the same way.

Basic functions allow computing diversity of a community. Data is simply a vector of probabilities (summing up to 1) or of abundances (integer values that are numbers of individuals). Calculate entropy with functions such as `Tsallis`, `Shannon`, `Simpson`, `Hurlbert` or `GenSimpson` and explicit diversity (i.e. effective number of species) with `Diversity` and others. By default, the best available estimator of diversity will be used, according to the data.

Communities can be simulated by `rCommunity`, explicitely declared as a species distribution (`as.AbdVector` or `as.ProbaVector`), and plotted.

Phylogenetic entropy and diversity can be calculated if a phylogenetic (or functional), ultrametric tree is provided. See `PhyloEntropy`, `Rao` for examples of entropy and `PhyloDiversity` to calculate phylodiversity, with the state-of-the-art estimation-bias correction. Similarity-based diversity is calculated with `Dqz`, based on a similarity matrix.

The simplest way to import data is to organize it into two text files. The first file should contain abundance data: the first column named `Species` for species names, and a column for each community.

The second file should contain the community weights in two columns. The first one, named `Communities` should contain their names and the second one, named `Weights`, their weights.

Files can be read and data imported by code such as:

```
Abundances <- read.csv(file="Abundances.csv", row.names = 1)
Weights <- read.csv(file="Weights.csv")
MC <- MetaCommunity(Abundances, Weights)
```

The last line of the code calls the `MetaCommunity` function to create an object that will be used by all metacommunity functions, such as `DivPart` (to partition diversity), `DivEst` (to partition diversity and calculate confidence interval of its estimation) or `DivProfile` (to compute diversity profiles).

A full documentation is available in the vignette. Type: `vignette("entropart")`. A quick introuction is in `vignette("introduction", "entropart")`.

Author(s)

Eric Marcon, Bruno Herault

References

Grabchak, M., Marcon, E., Lang, G., and Zhang, Z. (2017). The Generalized Simpson's Entropy is a Measure of Biodiversity. *Plos One*, 12(3): e0173305.

Marcon, E. (2015) Practical Estimation of Diversity from Abundance Data. *HAL* 01212435: 1-27.

Marcon, E. and Herault, B. (2015). entropart: An R Package to Measure and Partition Diversity. *Journal of Statistical Software*, 67(8): 1-26.

Marcon, E., Herault, B. (2015). Decomposing Phylodiversity. *Methods in Ecology and Evolution* 6(3): 333-339.

Marcon, E., Herault, B., Baraloto, C. and Lang, G. (2012). The Decomposition of Shannon's Entropy and a Confidence Interval for Beta Diversity. *Oikos* 121(4): 516-522.

Marcon, E., Scotti, I., Herault, B., Rossi, V. and Lang G. (2014). Generalization of the partitioning of Shannon diversity. *PLOS One* 9(3): e90289.

Marcon, E., Zhang, Z. and Herault, B. (2014). The decomposition of similarity-based diversity and its bias correction. *HAL* hal-00989454(version 3).

`AbdFreqCount` *Abundance Frequency Count of a Community*

Description

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

Usage

```
AbdFreqCount(Ns, Level = NULL, PCorrection="Chao2015", Unveiling="geom",
  RCorrection="Rarefy", CheckArguments = TRUE)
```

Arguments

`Ns`	A numeric vector containing species abundances.
`Level`	The level of interpolation or extrapolation. It may be an an arbitrary sample size (an integer) or a sample coverage (a number between 0 and 1).
`PCorrection`	A string containing one of the possible corrections to estimate a probability distribution in `as.ProbaVector`: `"Chao2015"` is the default value. Used only for extrapolation.
`Unveiling`	A string containing one of the possible unveiling methods to estimate the probabilities of the unobserved species in `as.ProbaVector`: `"geom"` (geometric: the unobserved species distribution is geometric) is the default value. Used only for extrapolation.
`RCorrection`	A string containing a correction recognized by `Richness` to evaluate the total number of species in `as.ProbaVector`. `"Rarefy"` is the default value to estimate the number of species such that the richness of the asymptotic distribution rarefied to the observed sample size equals the observed number of species in the data. Used only for extrapolation.
`CheckArguments`	Logical; 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 *et al.*, 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 `as.ProbaVector` and eq. (5) of Chao et al. (2014).

Value

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

References

Chao, A., Gotelli, N. J., Hsieh, T. C., Sander, E. L., Ma, K. H., Colwell, R. K., Ellison, A. M (2014). Rarefaction and extrapolation with Hill numbers: A framework for sampling and estimation in species diversity studies. *Ecological Monographs*, 84(1): 45-67.

Chao, A., Hsieh, T. C., Chazdon, R. L., Colwell, R. K., Gotelli, N. J. (2015) Unveiling the Species-Rank Abundance Distribution by Generalizing Good-Turing Sample Coverage Theory. *Ecology* 96(5): 1189-1201.

See Also

`PhyloEntropy`, `ChaoPD`

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest
#      and their taxonomy)
data(Paracou618)
# Ns is the vector of abundances of the first plot
Ns <- Paracou618.MC$Nsi[, 1]

# Return the abundance frequency count
(AbdFreqCount(Ns) -> afc)
plot(afc, xlab="Number of observations", ylab="Number of species")
lines(afc)
```

`Accumulation` *Diversity accumulation.*

Description

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

Usage

```
as.AccumCurve(x, y, low = NULL, high = NULL)
is.AccumCurve(x)
EntAC(Ns, q = 0, n.seq = seq_len(sum(Ns)), PCorrection="Chao2015", Unveiling="geom",
  RCorrection="Rarefy", NumberOfSimulations = 0, Alpha = 0.05,
  ShowProgressBar = TRUE, CheckArguments = TRUE)
DivAC(Ns, q = 0, n.seq = seq_len(sum(Ns)), PCorrection="Chao2015", Unveiling="geom",
  RCorrection="Rarefy", NumberOfSimulations = 0, Alpha = 0.05,
  ShowProgressBar = TRUE, CheckArguments = TRUE)
## S3 method for class 'AccumCurve'
plot(x, ..., main = NULL,
         xlab = "Sample Size", ylab = NULL, ylim = NULL,
         LineWidth = 2, ShadeColor = "grey75", BorderColor = "red")
## S3 method for class 'AccumCurve'
autoplot(object, ..., main = NULL,
         xlab = "Sample Size", ylab = NULL,
         ShadeColor = "grey75", alpha = 0.3, BorderColor = "red",
         col = "black",
         lty = 1,
         lwd = 0.5)
```

Arguments

`x`	An object. A numeric vector in `as.AccumCurve`.
`object`	An object.
`y`	A numeric vector.

`low`	A numeric vector.
`high`	A numeric vector.
`Ns`	A numeric vector containing species abundances.
`q`	A number: the order of diversity. Default is 1.
`n.seq`	A sequence of numbers. Accumulation will be calculated at each value.
`PCorrection`	A string containing one of the possible corrections to estimate a probability distribution in `as.ProbaVector`: `"Chao2015"` is the default value. Used only for extrapolation and `q` different from 0, 1, 2.
`Unveiling`	A string containing one of the possible unveiling methods to estimate the probabilities of the unobserved species in `as.ProbaVector`: `"geom"` (geometric: the unobserved species distribution is geometric) is the default value. Used only for extrapolation and `q` different from 0, 1, 2.
`RCorrection`	A string containing a correction recognized by `Richness` to evaluate the total number of species in `as.ProbaVector`. `"Rarefy"` is the default value to estimate the number of species such that the entropy of the asymptotic distribution rarefied to the observed sample size equals the observed entropy of the data. Used only for extrapolation and `q` different from 0, 1, 2. If `q` is 0 (extrapolation of richness), `"Rarefy"` is taken for `"Jackknife"`.
`NumberOfSimulations`	The number of Simulations to build confidence intervals.
`Alpha`	The risk level, 5% by default.
`...`	Additional arguments to be passed to `plot`. Unused elsewhere.
`main`	The main title of the plot. if `NULL` (by default), there is no title.
`xlab`	The X axis label, "Rank" by default.
`ylab`	The Y axis label. if `NULL` (by default), "Probability" or "Abundance" is chosen according to the object class.
`ylim`	The interval of y values plotted.
`LineWidth`	The width of the line that represents the actual profile.
`ShadeColor`	The color of the shaded confidence envelope.
`BorderColor`	The color of the bounds of the confidence envelope.
`alpha`	Opacity of the confidence enveloppe, between 0 and 1.
`col`	The color of the geom objects. See "Color Specification" in `par`.
`lty`	The type of the lines. See `lines`.
`lwd`	The width of the lines. See `lines`.
`ShowProgressBar`	If `TRUE` (default), a progress bar is shown.
`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

`DivAC` or `EntAC` estimate the diversity or entropy accumulation curve of a distribution. See `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. The asymptotic richess is adjusted so that the extrapolated part of the accumulation joins the observed value at the sample size.

`AccumCurve` objects include `EntAC` and `DivAC` objects for entropy and diversity accumulation. They generalize the classical Species Accumulation Curves (SAC) which are diversity accumulation of order $q=0$.

`as.AccumCurve` transforms two vectors (where `x` is the sammple size and `y` the accumulation) into an object of class `AccumCurve`.

`AccumCurve` objects can be plotted with either `plot` or `autoplot` `methods`.

Value

A `DivAC` or an `EntAC` object. Both are `AccumCurve` objects, which are a list:

`x`	The sample size.
`y`	The value of entropy or diversity.
`low`	The lower bound of the confidence envelope of the estimation.
`high`	The upper bound of the confidence envelope of the estimation.

Attibutes `"Size"` and `"Value"` contain the actual sample size and the corresponding diversity or entropy.

`AccumCurve` objects can be summarized and plotted.

References

Chao, A., Gotelli, N. J., Hsieh, T. C., Sander, E. L., Ma, K. H., Colwell, R. K., Ellison, A. M (2014). Rarefaction and extrapolation with Hill numbers: A framework for sampling and estimation in species diversity studies. *Ecological Monographs*, 84(1): 45-67.

See Also

`Tsallis`, `Diversity`

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest)
data(Paracou618)
# Ns is the total number of trees per species
Ns <- as.AbdVector(Paracou618.MC$Ns)
# Accumulation curve of Simpson's diversity
autoplot(DivAC(Ns, q=2))
```

`AllenH` *Phylogenetic Entropy of a Community*

Description

Calculates the phylogenetic diversity of order q of a probability vector.

Usage

```
AllenH(Ps, q = 1, PhyloTree, Normalize = TRUE, Prune = FALSE, CheckArguments = TRUE)
```

Arguments

`Ps`	A probability vector, summing to 1.
`q`	A number: the order of entropy. Default is 1.
`PhyloTree`	An object of class `hclust`, "phylo" (see `read.tree`), `phylog` or `PPtree`. The tree is not necessarily ultrametric.
`Normalize`	If `TRUE` (default), diversity is not affected by the height of the tree. If `FALSE`, it is proportional to the height of the tree.
`Prune`	What to do when somes species are in the tree but not in `Ps`? If `TRUE`, the tree is pruned to keep species of `Ps` only. The height of the tree may be changed if a pruned branch is related to the root. If `FALSE` (default), species with probability 0 are added in `Ps`.
`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

The phylogenetic entropy is calculated following Allen *et al.* (2009) for order $q = 1$ and Leinster and Cobold (2011) for other orders.The result is identical to the total entropy calculated by `PhyloEntropy` but it is much faster. A single value is returned instead of a `PhyloEntropy` object, and no bias correction is available.

The `Normalize` argument allows normalizing entropy by the height of the tree, similarly to `ChaoPD`.

Diversity can be calculated for non ultrametric trees following Leinster and Cobold (2011) even though the meaning of the result is not so clear.

Value

A named number equal the entropy of the community. The name is "None" to recall that no bias correction is available.

References

Allen, B., Kon, M. and Bar-Yam, Y. (2009). A New Phylogenetic Diversity Measure Generalizing the Shannon Index and Its Application to Phyllostomid Bats. *American Naturalist* 174(2): 236-243.

Leinster, T. and Cobbold, C. (2011). Measuring diversity: the importance of species similarity. *Ecology* 93(3): 477-489.

See Also

PhyloEntropy, ChaoPD

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest
#      and their taxonomy)
data(Paracou618)
# Ps is the vector of probabilities
Ps <- as.ProbaVector(Paracou618.MC$Ns)

# Calculate the phylogenetic Shannon diversity of the plot
AllenH(Ps, 1, Paracou618.Taxonomy, Normalize=TRUE)

# Calculate it using PhyloEntropy: more powerful but much slower is the tree has many periods
PhyloEntropy(Ps, 1, Paracou618.Taxonomy, Normalize=TRUE) -> phyE
summary(phyE)
```

AlphaDiversity *Reduced-bias alpha diversity of a metacommunity*

Description

Calculates the eeduced-bias total alpha diversity of order q of communities.

Usage

```
AlphaDiversity(MC, q = 1, Correction = "Best", Tree = NULL, Normalize = TRUE,
  Z = NULL, CheckArguments = TRUE)
```

Arguments

`MC`	A `MetaCommunity` object.
`q`	A number: the order of diversity. Default is 1 for Shannon diversity.
`Correction`	A string containing one of the possible corrections accepted by `AlphaEntropy` or `"None"` or `"Best"`, the default value.
`Tree`	An object of class `hclust`, "phylo" (see `read.tree`), `phylog` or `PPtree`. The tree must be ultrametric.
`Normalize`	If `TRUE` (default), diversity is not affected by the height of the tree. If `FALSE`, diversity is proportional to the height of the tree.
`Z`	A relatedness matrix, *i.e.* a square matrix whose terms are all positive, strictly positive on the diagonal. Generally, the matrix is a similarity matrix, *i.e.* the diagonal terms equal 1 and other terms are between 0 and 1.
`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

Entropy is calculated by `AlphaEntropy` and transformed into diversity.

Value

An `MCdiversity` object containing diversity values of each community and of the metacommunity.

References

Marcon, E., Scotti, I., Herault, B., Rossi, V. and Lang, G. (2014). Generalization of the partitioning of Shannon diversity. *PLOS One* 9(3): e90289.

Marcon, E., Herault, B. (2015). Decomposing Phylodiversity. *Methods in Ecology and Evolution* 6(3): 333-339.

Marcon, E., Zhang, Z. and Herault, B. (2014). The decomposition of similarity-based diversity and its bias correction. *HAL* hal-00989454(version 3).

See Also

`AlphaEntropy`

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest)
data(Paracou618)
# Calculate Simpson alpha diversity
summary(AlphaDiversity(Paracou618.MC, 2))
# Compare without correction
summary(AlphaDiversity(Paracou618.MC, 2, Correction = "None"))
# Estimate phylogenetic Simpson alpha diversity
summary(AlphaDiversity(Paracou618.MC, 2, Tree = Paracou618.Taxonomy) -> e)
plot(e)
```

`AlphaEntropy` *Reduced-bias alpha entropy of a metacommunity*

Description

Calculates the reduced-bias total alpha entropy of order q of communities.

Usage

```
AlphaEntropy(MC, q = 1, Correction = "Best", Tree = NULL, Normalize = TRUE,
  Z = NULL, CheckArguments = TRUE)
```

Arguments

`MC`	A `MetaCommunity` object.
`q`	A number: the order of diversity. Default is 1 for Shannon entropy.
`Correction`	A string containing one of the possible corrections accepted by the bias-corrected entropy function (see details) or `"None"` or `"Best"`, the default value.
`Tree`	An object of class `hclust`, "phylo" (see `read.tree`), `phylog` or `PPtree`. The tree must be ultrametric.
`Normalize`	If `TRUE` (default), the entropy returned by the function is normalized by the height of the tree (it is the weighted average value of the entropy in each slice). If `FALSE`, it is the unnormalized weighted sum of the results.
`Z`	A relatedness matrix, *i.e.* a square matrix whose terms are all positive, strictly positive on the diagonal. Generally, the matrix is a similarity matrix, *i.e.* the diagonal terms equal 1 and other terms are between 0 and 1.
`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

If `Tree` is not `NULL`, then phylogenetic entropy is calculated by `bcPhyloEntropy`; else, if `Z` is not `NULL`, then similarity-based entropy is calculated by `bcHqz`; else, neutral entropy is calculated by `bcTsallis`.

The alpha entropy of each community is calculated and summed according to community weights.

The possible corrections are detailed in `Tsallis`.

Value

An `MCentropy` object containing entropy values of each community and of the metacommunity.

References

Marcon, E., Scotti, I., Herault, B., Rossi, V. and Lang, G. (2014). Generalization of the partitioning of Shannon diversity. *PLOS One* 9(3): e90289.

Marcon, E., Herault, B. (2015). Decomposing Phylodiversity. *Methods in Ecology and Evolution* 6(3): 333-339.

Marcon, E., Zhang, Z. and Herault, B. (2014). The decomposition of similarity-based diversity and its bias correction. *HAL* hal-00989454(version 3).

See Also

`bcTsallis`

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest)
data(Paracou618)
# Calculate Simpson alpha entropy
summary(AlphaEntropy(Paracou618.MC, 2))
# Compare without correction
summary(AlphaEntropy(Paracou618.MC, 2, Correction = "None"))
# Estimate phylogenetic Simpson alpha entropy
summary(AlphaEntropy(Paracou618.MC, 2, Tree = Paracou618.Taxonomy) -> e)
plot(e)
```

`BetaDiversity` *Reduced-bias beta diversity of a metacommunity*

Description

Calculates the reduced-bias beta diversity of order q between communities.

Usage

```
BetaDiversity(MC, q = 1, Correction = "Best", Tree = NULL, Normalize = TRUE,
  Z = NULL, CheckArguments = TRUE)
```

Arguments

`MC`	A `MetaCommunity` object.
`q`	A number: the order of diversity. Default is 1 for Shannon diversity.
`Correction`	A string containing one of the possible corrections accepted by `bcTsallisBeta` or `"None"` or `"Best"`, the default value.
`Tree`	An object of class `hclust`, "phylo" (see `read.tree`), `phylog` or `PPtree`. The tree must be ultrametric.
`Normalize`	If `TRUE` (default), diversity is not affected by the height of the tree. If `FALSE`, diversity is proportional to the height of the tree.
`Z`	A relatedness matrix, *i.e.* a square matrix whose terms are all positive, strictly positive on the diagonal. Generally, the matrix is a similarity matrix, *i.e.* the diagonal terms equal 1 and other terms are between 0 and 1.
`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

Entropy is calculated by `BetaEntropy` and transformed into diversity.

Diversity values of communities are not defined: community entropies are averaged to obtain the metacommunity entropy wich is transformed into diversity (Marcon et al., 2014).

Value

An `MCdiversity` object containing diversity value of the metacommunity.

References

Marcon, E., Scotti, I., Herault, B., Rossi, V. and Lang, G. (2014). Generalization of the partitioning of Shannon diversity. *PLOS One* 9(3): e90289.

Marcon, E., Herault, B. (2015). Decomposing Phylodiversity. *Methods in Ecology and Evolution* 6(3): 333-339.

Marcon, E., Zhang, Z. and Herault, B. (2014). The decomposition of similarity-based diversity and its bias correction. *HAL* hal-00989454(version 3).

See Also

`BetaEntropy`

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest)
data(Paracou618)
# Estimate Shannon beta diversity
summary(BetaDiversity(Paracou618.MC, 1))
# Compare without correction
summary(BetaDiversity(Paracou618.MC, 1, Correction = "None"))
# Estimate phylogenetic Shannon beta diversity
summary(BetaDiversity(Paracou618.MC, 1, Tree = Paracou618.Taxonomy) -> e)
```

`BetaEntropy` *Reduced-bias beta entropy of a metacommunity*

Description

Calculates the reduced-bias beta entropy of order q between communities.

Usage

```
BetaEntropy(MC, q = 1, Correction = "Best", Tree = NULL, Normalize = TRUE,
  Z = NULL, CheckArguments = TRUE)
```

Arguments

`MC`	A `MetaCommunity` object.
`q`	A number: the order of diversity. Default is 1 for Shannon entropy.
`Correction`	A string containing one of the possible corrections accepted by the bias-corrected entropy function (see details) or `"None"` or `"Best"`, the default value.
`Tree`	An object of class `hclust`, "phylo" (see `read.tree`), `phylog` or `PPtree`. The tree must be ultrametric.

`Normalize`	If `TRUE` (default), the entropy returned by the function is normalized by the height of the tree (it is the weighted average value of the entropy in each slice). If `FALSE`, it is the unnormalized weighted sum of the results.
`Z`	A relatedness matrix, *i.e.* a square matrix whose terms are all positive, strictly positive on the diagonal. Generally, the matrix is a similarity matrix, *i.e.* the diagonal terms equal 1 and other terms are between 0 and 1.
`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

If `Tree` is not `NULL`, then phylogenetic entropy is calculated by `bcPhyloBetaEntropy`; else, if `Z` is not `NULL`, then similarity-based entropy is calculated by `bcHqzBeta`; else, neutral entropy is calculated by `bcTsallisBeta`.

The reduced-bias beta entropy of each community is calculated and summed according to community weights.

Note that beta entropy is related to alpha entropy (if q is not 1) and cannot be compared accross communities (Jost, 2007). Do rather calculate the `BetaDiversity` of the metacommunity.

Value

An `MCentropy` object containing entropy values of each community and of the metacommunity.

References

Marcon, E., Scotti, I., Herault, B., Rossi, V. and Lang, G. (2014). Generalization of the partitioning of Shannon diversity. *PLOS One* 9(3): e90289.

Marcon, E., Herault, B. (2015). Decomposing Phylodiversity. *Methods in Ecology and Evolution* 6(3): 333-339.

Marcon, E., Zhang, Z. and Herault, B. (2014). The decomposition of similarity-based diversity and its bias correction. *HAL* hal-00989454(version 3).

See Also

`bcTsallisBeta`, `BetaDiversity`

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest)
data(Paracou618)
# Estimate Shannon beta entropy
summary(BetaEntropy(Paracou618.MC, 1))
# Compare without correction
summary(BetaEntropy(Paracou618.MC, 1, Correction = "None"))
# Estimate phylogenetic Shannon beta entropy
summary(BetaEntropy(Paracou618.MC, 1, Tree = Paracou618.Taxonomy) -> e)
plot(e)
```

`ChaoPD` *Phylogenetic Diversity of a Community*

Description

Calculates the phylogenetic diversity of order q of a probability vector.

Usage

```
ChaoPD(Ps, q = 1, PhyloTree, Normalize = TRUE, Prune = FALSE, CheckArguments = TRUE)
```

Arguments

`Ps`	A probability vector, summing to 1.
`q`	A number: the order of diversity. Default is 1.
`PhyloTree`	An object of class `hclust`, "phylo" (see `read.tree`), `phylog` or `PPtree`. The tree is not necessarily ultrametric.
`Normalize`	If `TRUE` (default), diversity is not affected by the height of the tree. If `FALSE`, it is proportional to the height of the tree.
`Prune`	What to do when somes species are in the tree but not in `Ps`? If `TRUE`, the tree is pruned to keep species of `Ps` only. The height of the tree may be changed if a pruned branch is related to the root. If `FALSE` (default), species with probability 0 are added in `Ps`.
`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

The phylogenetic diversity is calculated following Chao *et al.* (2010). The result is identical to the total diversity calculated by `PhyloDiversity` but it is much faster. A single value is returned instead of a `PhyloDiversity` object, and no bias correction is available.

The `Normalize` arguments allows calculating either $^q\bar{D}(T)$ (if `TRUE`) or $^qPD(T)$ if `FALSE`.

Diversity can be calculated for non ultrametric trees following Chao *et al.* (2010) even though the meaning of the result is not so clear (Leinster and Cobold, 2011).

Value

A named number equal the diversity of the community. The name is "None" to recall that no bias correction is available.

References

Chao, A., Chiu, C.-H. and Jost, L. (2010). Phylogenetic diversity measures based on Hill numbers. *Philosophical Transactions of the Royal Society B* 365(1558): 3599-609.

Leinster, T. and Cobbold, C. (2011). Measuring diversity: the importance of species similarity. *Ecology* 93(3): 477-489.

See Also

PhyloDiversity, AllenH

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest
#      and their taxonomy)
data(Paracou618)
# Ps is the vector of probabilities
Ps <- Paracou618.MC$Ps

# Calculate the phylogenetic Simpson diversity of the plot
(ChaoPD(Paracou618.MC$Ps, 2, Paracou618.Taxonomy, Normalize=TRUE))

# Calculate it using PhyloDiversity
# (more powerful but much slower if the tree has many periods)
PhyloDiversity(Paracou618.MC$Ps, 2, Paracou618.Taxonomy, Normalize=TRUE) -> phyD
summary(phyD)
```

`CommunityProfile` *Diversity or Entropy Profile of a community*

Description

Calculates the diversity or entropy profile of a community, applying a community function to a vector of orders.

Usage

```
CommunityProfile(FUN, NorP, q.seq = seq(0, 2, 0.1),
    NumberOfSimulations = 0, Alpha = 0.05, BootstrapMethod = "Chao2015",
    size = 1, ..., ShowProgressBar = TRUE, CheckArguments = TRUE)
as.CommunityProfile(x, y, low = NULL, high = NULL, mid = NULL)
is.CommunityProfile(x)
## S3 method for class 'CommunityProfile'
plot(x, ..., main = NULL,
          xlab = "Order of Diversity", ylab = "Diversity", ylim = NULL,
          LineWidth = 2, ShadeColor = "grey75", BorderColor = "red")
## S3 method for class 'CommunityProfile'
autoplot(object, ..., main = NULL,
          xlab = "Order of Diversity", ylab = "Diversity",
          ShadeColor = "grey75", alpha = 0.3, BorderColor = "red",
          col = "black",
          lty = 1,
          lwd = 0.5)
CEnvelope(Profile, LineWidth = 2, ShadeColor = "grey75", BorderColor = "red", ...)
```

Arguments

`FUN`	The function to be applied to each value of $q.seq$. Any function accepting a numeric vector (or a two-column matrix) and a number as first two arguments and an argument named `CheckArguments` is acceptable (other arguments of the functions are passed by `...`). See *Details* for useful entropy and diversity functions and *Examples* for an ad-hoc one.
`NorP`	A numeric vector. Contains either abundances or probabilities.
`q.seq`	A numeric vector: the sequence of diversity orders to address. Default is from 0 to 2.
`NumberOfSimulations`	The number of simulations to run, 0 by default.
`Alpha`	The risk level, 5% by default.
`BootstrapMethod`	The method used to obtain the probabilities to generate bootstrapped communities from observed abundances. See `rCommunity`.
`size`	The size of simulated communities used to compute the bootstrap confidence envelope. 1 (default) means that the actual size must be used.
`object`	An object.
`x`	An object to be tested or plotted or the vector of orders of community profiles in `as.CommunityProfile`.
`y`	Entropy or diversity values of each order, corresponding to `x` values.
`low`	Entropy or diversity lower bound of the confidence envelope, corresponding to `x` values.
`high`	Entropy or diversity higher bound of the confidence envelope, corresponding to `x` values.
`mid`	Entropy or diversity center value (usually the mean) of the confidence envelope, corresponding to `x` values.
`Profile`	An `CommunityProfile` to be plotted.
`...`	Additional arguments to be passed to `FUN` in `CommunityProfile`, to `plot` in `plot.CommunityProfile` or to `lines` in `CEnvelope`.
`main`	The main title of the plot.
`xlab`	The x axis label of the plots.
`ylab`	The y axis label of the plot.
`ylim`	The interval of y values plotted.
`LineWidth`	The width of the line that represents the actual profile.
`ShadeColor`	The color of the shaded confidence envelope.
`BorderColor`	The color of the bounds of the confidence envelope.
`alpha`	Opacity of the confidence enveloppe, between 0 and 1.
`col`	The color of the geom objects. See "Color Specification" in `par`.
`lty`	The type of the lines. See `lines`.
`lwd`	The width of the lines. See `lines`.

`ShowProgressBar`	If `TRUE` (default), a progress bar is shown.
`CheckArguments`	Logical; 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 `CommunityProfile` is used to calculate diversity or entropy profiles based on community functions such as `Tsallis` or `ChaoPD`. The first two arguments of the function must be a probability or abundance vector and a number (q). Additional arguments cannot be checked. Unexpected results may be returned if `FUN` is not used properly.

If `NumberOfSimulations` is greater than 0, a bootstrap confidence interval is produced by simulating communities with `rCommunity` and calculating their profiles. The size of those communities may be that of the actual community or specified by `size`. 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 if ‘size = 1‘ so that their mean is that of the actual community. Else, it is assumed that the bias is of interest and must not be corrected.

`CommunityProfile` objects can be plotted. They can also be added to the current plot by `CEnvelope`.

Value

A `CommunityProfile`, which is a list:

`x`	The order q values
`y`	The entropy or diversity values returned by `FUN`
`low`	The lower bound of the confidence interval
`high`	The upper bound of the confidence interval

Author(s)

Eric Marcon <Eric.Marcon@agroparistech.fr>, Bruno Herault <Bruno.Herault@cirad.fr>

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest)
data(Paracou618)
# Plot diversity estimated without bias correction
plot(CommunityProfile(Diversity, Paracou618.MC$Ps, seq(0, 2, 0.2)),
lty=3, ylim=c(50, 350))
# Estimate diversity, with a condidence envelope
# (only 10 simulations to save time, should be 1000)
Profile <- CommunityProfile(Diversity, as.AbdVector(Paracou618.MC$Ns),
  seq(0, 2, 0.2), Correction="UnveilJ", NumberOfSimulations=10)
# Complete the plot, and add the legend
CEnvelope(Profile, main="Paracou Plots Diversity")
legend("topright", c("Bias Corrected", "Biased"), lty=c(1,3), inset=0.01)

# Advanced use with beta-diversity functions :
```

```
# Profile of the beta entropy of the first community of Paracou618.
# Observed and expected probabilities are bound into a 2-column matrix
# An intermediate function is necessary to separate them before calling TsallisBeta
# The CheckArguments is mandatory but does not need to be set: CommunityProfile() sets it to FALSE
CommunityProfile(function(PandPexp, q, CheckArguments)
  {TsallisBeta(PandPexp[, 1], PandPexp[, 2], q)},
  NorP=cbind(Paracou618.MC$Psi[, 1], Paracou618.MC$Ps), q.seq=seq(0, 2, 0.2))
```

`Coverage` *Sample coverage of a community*

Description

`"Coverage"` calculates an estimator of the sample coverage of a community described by its abundance vector. `"Coverage2Size"` estimates the sample size corresponding to the chosen sample coverage.

Usage

```
Coverage(Ns, Estimator = "Best", Level = NULL, CheckArguments = TRUE)
Coverage2Size(Ns, SampleCoverage, CheckArguments = TRUE)
```

Arguments

`Ns`	A numeric vector containing species abundances.
`Estimator`	A string containing one of the possible estimators: `"ZhangHuang"`, `"Chao"`, `"Turing"`, `"Good"`. `"Best"` is for `"ZhangHuang"`.
`Level`	The level of interpolation or extrapolation, i.e. an abundance.
`SampleCoverage`	The target sample coverage.
`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

The sample coverage C of a community is the total probability of occurence 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 (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. `"Coverage2Size"` returns a number equal to the sample size corresponding to the chosen sample coverage.

References

Chao, A., Lee, S.-M. and Chen, T.-C. (1988). A generalized Good's nonparametric coverage estimator. *Chinese Journal of Mathematics* 16: 189-199.

Chao, A. and Shen, T.-J. (2010). *Program SPADE: Species Prediction And Diversity Estimation. Program and user's guide*. CARE, Hsin-Chu, Taiwan.

Chao, A., Gotelli, N. J., Hsieh, T. C., Sander, E. L., Ma, K. H., Colwell, R. K., Ellison, A. M (2014). Rarefaction and extrapolation with Hill numbers: A framework for sampling and estimation in species diversity studies. *Ecological Monographs*, 84(1): 45-67.

Good, I. J. (1953). On the Population Frequency of Species and the Estimation of Population Parameters. *Biometrika* 40(3/4): 237-264.

Zhang, Z. and Huang, H. (2007). Turing's formula revisited. *Journal of Quantitative Linguistics* 14(2-3): 222-241.

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest)
data(Paracou618)
# Ns is the vector of abundances of the metacommunity
Ns <- Paracou618.MC$Ns
# Calculate the sample coverage of the metacommunity
Coverage(Ns)    # Stored in Paracou618.SampleCoverage
```

`Diversity` *Hill number of a community*

Description

Calculates the HCDT (generalized) diversity of order q of a probability vector.

Usage

```
Diversity(NorP, q = 1, ...)
bcDiversity(Ns, q = 1, Correction = "Best", CheckArguments = TRUE)
## S3 method for class 'ProbaVector'
Diversity(NorP, q = 1, ...,
  CheckArguments = TRUE, Ps = NULL)
## S3 method for class 'AbdVector'
Diversity(NorP, q = 1, Correction = "Best", Level = NULL,
  PCorrection="Chao2015", Unveiling="geom", RCorrection="Rarefy", ...,
  CheckArguments = TRUE, Ns = NULL)
```

```
## S3 method for class 'integer'
Diversity(NorP, q = 1, Correction = "Best", Level = NULL,
  PCorrection="Chao2015", Unveiling="geom", RCorrection="Rarefy", ...,
  CheckArguments = TRUE, Ns = NULL)
## S3 method for class 'numeric'
Diversity(NorP, q = 1, Correction = "Best", Level = NULL,
  PCorrection="Chao2015", Unveiling="geom", RCorrection="Rarefy", ...,
  CheckArguments = TRUE, Ps = NULL, Ns = NULL)
```

Arguments

`Ps`	A probability vector, summing to 1.
`Ns`	A numeric vector containing species abundances.
`NorP`	A numeric vector, an integer vector, an abundance vector (`AbdVector`) or a probability vector (`ProbaVector`). Contains either abundances or probabilities.
`q`	A number: the order of diversity. Default is 1.
`Correction`	A string containing one of the possible asymptotic estimators: `"None"` (no correction), `"ChaoShen"`, `"GenCov"`, `"Grassberger"`, `"Holste"`, `"Bonachela"`, `"ZhangGrabchak"`, or `"ChaoJost"`, `"Marcon"`, `"UnveilC"`, `"UnveiliC"`, `"UnveilJ"` or `"Best"`, the default value. Currently, `"Best"` is `"UnveilJ"`.
`Level`	The level of interpolation or extrapolation. It may be an a chosen sample size (an integer) or a sample coverage (a number between 0 and 1).
`PCorrection`	A string containing one of the possible corrections to estimate a probability distribution in `as.ProbaVector`: `"Chao2015"` is the default value. Used only for extrapolation.
`Unveiling`	A string containing one of the possible unveiling methods to estimate the probabilities of the unobserved species in `as.ProbaVector`: `"geom"` (the unobserved species distribution is geometric) is the default value. Used only for extrapolation.
`RCorrection`	A string containing a correction recognized by `Richness` to evaluate the total number of species in `as.ProbaVector`. `"Rarefy"` is the default value to estimate the number of species such that the diversity of the asymptotic distribution rarefied to the observed sample size equals the observed diversity of the data. Used only for extrapolation.
`...`	Additional arguments. Unused.
`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

`Diversity` calls `Tsallis` to calculate entropy and transforms it into diversity by calculating its deformed exponential.

Bias correction requires the number of individuals to estimate sample `Coverage`. See `Tsallis` for details.

The functions are designed to be used as simply as possible. `Diversity` is a generic method. If its first argument is an abundance vector, an integer vector or a numeric vector which does not sum to 1, the bias corrected function `bcDiversity` is called.

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 `Tsallis` for details.

Value

A named number equal to the calculated diversity. The name is that of the bias correction used.

References

Chao, A., Gotelli, N. J., Hsieh, T. C., Sander, E. L., Ma, K. H., Colwell, R. K., Ellison, A. M (2014). Rarefaction and extrapolation with Hill numbers: A framework for sampling and estimation in species diversity studies. *Ecological Monographs*, 84(1): 45-67.

Marcon, E., Scotti, I., Herault, B., Rossi, V. and Lang, G. (2014). Generalization of the partitioning of Shannon diversity. *PLOS One* 9(3): e90289.

See Also

`Tsallis`, `expq`, `AbdVector`, `ProbaVector`

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest)
data(Paracou618)
# Ns is the total number of trees per species
Ns <- as.AbdVector(Paracou618.MC$Ns)
# Species probabilities
Ps <- as.ProbaVector(Paracou618.MC$Ns)
# Whittaker plot
plot(Ns)
# Calculate diversity of order 1, i.e. Shannon's diversity
Diversity(Ps, q=1)
# Calculate it with estimation bias correction (asymptotic estimator)
Diversity(Ns, q=1)
# Extrapolate it up to 99.9% sample coverage (close to the asymptotic estimator)
Diversity(Ns, q=1, Level=0.999)
# Rarefy it to half the sample size
Diversity(Ns, q=1, Level=sum(Ns)/2)
```

`DivEst` *Diversity Estimation of a metacommunity*

Description

Estimates diversity of a metacommunity.

Usage

```
DivEst(q = 0, MC, Biased = TRUE, Correction = "Best", Tree = NULL,
  Normalize = TRUE, Z = NULL, Simulations = 100,
  ShowProgressBar = TRUE, CheckArguments = TRUE)
is.DivEst(x)
## S3 method for class 'DivEst'
plot(x, ..., main = NULL, Which = "All",
  Quantiles = c(0.025, 0.975), colValue = "red", lwdValue = 2, ltyValue = 2,
  colQuantiles = "black", lwdQuantiles = 1, ltyQuantiles = 2)
## S3 method for class 'DivEst'
autoplot(object, ..., main = NULL, Which = "All",
  labels = NULL, font.label = list(size=11, face="plain"),
          Quantiles = c(0.025, 0.975), colValue = "red",
          colQuantiles = "black", ltyQuantiles = 2)
## S3 method for class 'DivEst'
summary(object, ...)
```

Arguments

`q`	A number: the order of diversity.
`MC`	A `MetaCommunity` object.
`Biased`	Logical; if `FALSE`, a bias correction is appplied.
`Correction`	A string containing one of the possible corrections. The correction must be accepted by `DivPart`. `"Best"` is the default value.
`Tree`	An object of class `hclust`, "phylo" (see `read.tree`), `phylog` or `PPtree`. The tree must be ultrametric.
`Normalize`	If `TRUE` (default), diversity is not affected by the height of the tree.. If `FALSE`, diversity is proportional to the height of the tree.
`Z`	A relatedness matrix, *i.e.* a square matrix whose terms are all positive, strictly positive on the diagonal. Generally, the matrix is a similarity matrix, *i.e.* the diagonal terms equal 1 and other terms are between 0 and 1.
`Simulations`	The number of simulations to build confidence intervals.
`ShowProgressBar`	If `TRUE` (default), a progress bar is shown.
`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.
`x`	An object to be tested or plotted.
`main`	The title of the plot.
`Which`	May be `"Alpha"`, `"Beta"` or `"Gamma"` to respectively plot the metacommunity's alpha, beta or gamma diversity. If `"All"` (default), all three plots are shown.
`labels`	Vector of labels to be added to multiple plots. `"auto"` is the same as `c("a", "b", "c", "d")`.
`font.label`	A list of arguments to customize labels. See `ggarrange`.
`object`	A `MCdiversity` object to be summarized or plotted.

`Quantiles`	A vector containing the quantiles of interest.
`colValue`	The color of the line representing the real value on the plot.
`lwdValue`	The width of the line representing the real value on the plot.
`ltyValue`	The line type of the line representing the real value on the plot.
`colQuantiles`	The color of the lines representing the quantiles on the plot.
`lwdQuantiles`	The width of the lines representing the quantiles on the plot.
`ltyQuantiles`	The line type of the lines representing the quantiles on the plot.
`...`	Additional arguments to be passed to the generic methods.

Details

`Divest` estimates the diversity of the metacommunity and partitions it into alpha and beta components.

If `Tree` is provided, the phylogenetic diversity is calculated else if `Z` is not `NULL`, then similarity-based entropy is calculated.

Bootstrap confidence intervals are calculated by drawing simulated communities from a multinomial distribution following the observed frequencies (Marcon et al, 2012; 2014).

Value

A `Divest` object which is a `DivPart` object with an additional item in its list:

`SimulatedDiversity`	A matrix containing the simulated values of alpha, beta and gamma diversity.

`Divest` objects can be summarized and plotted.

Author(s)

Eric Marcon <Eric.Marcon@agroparistech.fr>, Bruno Herault <Bruno.Herault@cirad.fr>

References

Marcon, E., Herault, B., Baraloto, C. and Lang, G. (2012). The Decomposition of Shannon's Entropy and a Confidence Interval for Beta Diversity. *Oikos* 121(4): 516-522.

Marcon, E., Scotti, I., Herault, B., Rossi, V. and Lang, G. (2014). Generalization of the partitioning of Shannon diversity. *PLOS One* 9(3): e90289.

Marcon, E., Herault, B. (2015). Decomposing Phylodiversity. *Methods in Ecology and Evolution* 6(3): 333-339.

See Also

`DivPart`

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest)
data(Paracou618)
# Estimate Shannon diversity.
Estimation <- DivEst(q = 1, Paracou618.MC, Biased = FALSE, Correction = "UnveilJ",
Simulations = 20)
plot(Estimation)
summary(Estimation)
```

`DivPart` *Diversity Partition of a metacommunity*

Description

Partitions the diversity of a metacommunity into alpha and beta components.

Usage

```
DivPart(q = 1, MC, Biased = TRUE, Correction = "Best", Tree = NULL,
  Normalize = TRUE, Z = NULL, CheckArguments = TRUE)
is.DivPart(x)
## S3 method for class 'DivPart'
plot(x, ...)
## S3 method for class 'DivPart'
autoplot(object, col = "grey35",
  border = NA, ...)
## S3 method for class 'DivPart'
summary(object, ...)
```

Arguments

`q`	A number: the order of diversity. Default is 1.
`MC`	A `MetaCommunity` object.
`Biased`	Logical; if `FALSE`, a bias correction is appplied.
`Correction`	A string containing one of the possible corrections. The correction must be accepted by `AlphaEntropy`, `BetaEntropy` and `GammaEntropy`. `"Best"` is the default value.
`Tree`	An object of class `hclust`, "phylo" (see `read.tree`), `phylog` or `PPtree`. The tree must be ultrametric.
`Normalize`	If `TRUE` (default), diversity is not affected by the height of the tree. If `FALSE`, diversity is proportional to the height of the tree.
`Z`	A relatedness matrix, *i.e.* a square matrix whose terms are all positive, strictly positive on the diagonal. Generally, the matrix is a similarity matrix, *i.e.* the diagonal terms equal 1 and other terms are between 0 and 1.

`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.
`x`	An object to be tested or plotted.
`object`	A `MCdiversity` object to be summarized or plotted.
`col`	The color used to fill the bars. See "Color Specification" in `par`.
`border`	The color of the borders around the bars. See `rect`.
`...`	Additional arguments to be passed to the generic methods.

Details

`DivPart` partitions the diversity of the metacommunity into alpha and beta components. It supports estimation-bias correction.

If `Tree` is provided, the phylogenetic diversity is calculated else if `Z` is not `NULL`, then similarity-based entropy is calculated.

Beta diversity/entropy is calculated from Gamma and Alpha when bias correction is required, so community values are not available.

Value

A `DivPart` object. It is a list:

`MetaCommunity`	The name of the `MetaCommunity` object containing inventory data.
`Order`	The value of q.
`Biased`	Logical. If `FALSE`, bias corrected values of diversity have been computed.
`Correction`	The estimation bias correction used to calculate diversity.
`Method`	The method used to calculate entropy ("HCDT", "Similarity-based").
`Tree`	The phylogenetic or functional tree used to calculate phylodiversity.
`Normalized`	Logical. Indicates whether phylodiversity is normalized or proportional to the height of the tree.
`Z`	The matrix used to calculate similarity-based entropy.
`TotalAlphaDiversity`	The alpha diversity of communities.
`TotalBetaDiversity`	The beta diversity of communities.
`GammaDiversity`	The gamma diversity of the metacommunity.
`CommunityAlphaDiversities`	A vector containing the alpha diversity of each community.
`TotalAlphaEntropy`	The alpha entropy of communities.
`TotalBetaEntropy`	The beta entropy of communities.
`GammaEntropy`	The gamma entropy of the metacommunity.
`CommunityAlphaEntropies`	A vector containing the alpha entropy of each community.

`DivPart` objects can be summarized and plotted.

Author(s)

Eric Marcon <Eric.Marcon@agroparistech.fr>, Bruno Herault <Bruno.Herault@cirad.fr>

References

Marcon, E., Herault, B., Baraloto, C. and Lang, G. (2012). The Decomposition of Shannon's Entropy and a Confidence Interval for Beta Diversity. *Oikos* 121(4): 516-522.

Marcon, E., Scotti, I., Herault, B., Rossi, V. and Lang, G. (2014). Generalization of the partitioning of Shannon diversity. *PLOS One* 9(3): e90289.

Marcon, E., Herault, B. (2015). Decomposing Phylodiversity. *Methods in Ecology and Evolution* 6(3): 333-339.

Marcon, E., Zhang, Z. and Herault, B. (2014). The decomposition of similarity-based diversity and its bias correction. *HAL* hal-00989454(version 3).

See Also

`DivProfile`

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest)
data(Paracou618)
# Estimate Shannon diversity.
summary(DivPart(q = 1, Paracou618.MC, Biased = FALSE) -> dp)
plot(dp)
```

`DivProfile` *Diversity Profile of a metacommunity*

Description

Calculate the diversity profiles (alpha, beta, gamma) of a metacommunity.

Usage

```
DivProfile(q.seq = seq(0, 2, 0.1), MC, Biased = TRUE, Correction = "Best",
  Tree = NULL, Normalize = TRUE, Z = NULL,
  NumberOfSimulations = 0, Alpha = 0.05,
  ShowProgressBar = TRUE, CheckArguments = TRUE)
is.DivProfile(x)
## S3 method for class 'DivProfile'
plot(x, ..., main = NULL, xlab = "Order of Diversity",
  ylab = NULL, Which = "All",
  LineWidth = 2, ShadeColor = "grey75", BorderColor = "red")
## S3 method for class 'DivProfile'
autoplot(object, ..., main = NULL, xlab = "Order of Diversity",
```

```
  ylab = NULL, Which = "All", ShadeColor = "grey75", alpha = 0.3, BorderColor = "red",
   labels = NULL, font.label = list(size=11, face="plain"),
   col = "black",
   lty = 1,
   lwd = 0.5)
## S3 method for class 'DivProfile'
summary(object, ...)
```

Arguments

`q.seq`	A numeric vector.
`MC`	A `MetaCommunity` object.
`Biased`	Logical; if `FALSE`, a bias correction is appplied.
`Correction`	A string containing one of the possible corrections. The correction must be accepted by `AlphaEntropy`, `BetaEntropy` and `GammaEntropy`. `"Best"` is the default value.
`Tree`	An object of class `hclust`, "phylo" (see `read.tree`), `phylog` or `PPtree`. The tree must be ultrametric.
`Normalize`	If `TRUE` (default), diversity is not affected by the height of the tree. If `FALSE`, diversity is proportional to the height of the tree.
`Z`	A relatedness matrix, *i.e.* a square matrix whose terms are all positive, strictly positive on the diagonal. Generally, the matrix is a similarity matrix, *i.e.* the diagonal terms equal 1 and other terms are between 0 and 1.
`NumberOfSimulations`	The number of simulations to run, 0 by default.
`Alpha`	The risk level, 5% by default.
`ShowProgressBar`	If `TRUE` (default), a progress bar is shown.
`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.
`x`	An object to be tested or plotted.
`main`	The main title of the plot. Ignored if `Which = "All"`.
`xlab`	The x axis label of the plots.
`ylab`	The y axis label of the plot. Ignored if `Which = "All"`.
`Which`	May be `"Communities"`, `"Alpha"`, `"Beta"` or `"Gamma"` to respectively plot the alpha diversity of communities or the metacommunity's alpha, beta or gamma diversity. If `"All"` (default), all four plots are shown.
`LineWidth`	The width of the line that represents the actual profile.
`ShadeColor`	The color of the shaded confidence envelope.
`BorderColor`	The color of the bounds of the confidence envelope.
`alpha`	Opacity of the confidence enveloppe, between 0 and 1.
`labels`	Vector of labels to be added to multiple plots. `"auto"` is the same as `c("a", "b", "c", "d")`.

`font.label`	A list of arguments to customize labels. See `ggarrange`.
`col`	The color of the geom objects. See "Color Specification" in `par`.
`lty`	The type of the lines. See `lines`.
`lwd`	The width of the lines. See `lines`.
`object`	A `MCdiversity` object to be summarized or plotted.
`...`	Additional arguments to be passed to the generic methods.

Details

If `Tree` is provided, the phylogenetic diversity is calculated.

`DivPart` partitions the diversity of the metacommunity into alpha and beta components. It supports estimation-bias correction.

If `Tree` is provided, the phylogenetic diversity is calculated else if `Z` is not `NULL`, then similarity-based entropy is calculated.

Beta diversity/entropy is calculated from Gamma and Alpha when bias correction is required, so community values are not available.

If `NumberOfSimulations` is greater than 0, a bootstrap confidence interval is produced by simulating communities from a multinomial distribution following the observed frequencies (Marcon et al, 2012; 2014) and calculating their profiles.

Value

A `DivProfile` object. It is a list:

`MetaCommunity`	The name of the `MetaCommunity` object containing inventory data.
`Order`	A vector containing the values of q.
`Biased`	Logical. If `FALSE`, bias corrected values of diversity have been computed.
`Correction`	The estimation bias correction used to calculate diversity. Usually a string, but it may be a list if different corrections have been used in the estimation of phylodiversity.
`Method`	The method used to calculate entropy ("HCDT", "Similarity-based").
`Tree`	The phylogenetic or functional tree used to calculate phylodiversity.
`Normalized`	Logical. Indicates whether phylodiversity is normalized or proportional to the height of the tree.
`Z`	The matrix used to calculate similarity-based entropy.
`CommunityAlphaDiversities`	A matrix containing the alpha diversity of each community.
`TotalAlphaDiversity`	A vector containing the alpha diversity of communities for each order.
`BetaDiversity`	A vector containing the beta diversity of communities for each order.
`GammaDiversity`	A vector containing the gamma diversity of the metacommunity for each order.
`CommunityAlphaEntropies`	A matrix containing the alpha entropy of each community.

`TotalAlphaEntropy`	A vector containing the alpha entropy of communities for each order.
`BetaEntropy`	A vector containing the beta entropy of communities for each order.
`GammaEntropy`	A vector containing the gamma entropy of the metacommunity for each order.
`Confidence envelopes`	Total Alpha, Beta and Gamma Entropy and Diversity may come with a confidence envelope whose value is stored in twelve more vectors named suffixed Low or High, such as `GammaEntropyLow`

`DivProfile` objects can be summarized and plotted.

Author(s)

Eric Marcon <Eric.Marcon@agroparistech.fr>, Bruno Herault <Bruno.Herault@cirad.fr>

References

Marcon, E., Herault, B., Baraloto, C. and Lang, G. (2012). The Decomposition of Shannon's Entropy and a Confidence Interval for Beta Diversity. *Oikos* 121(4): 516-522.

Marcon, E., Scotti, I., Herault, B., Rossi, V. and Lang, G. (2014). Generalization of the partitioning of Shannon diversity. *PLOS One* 9(3): e90289.

Marcon, E., Herault, B. (2015). Decomposing Phylodiversity. *Methods in Ecology and Evolution* 6(3): 333-339.

See Also

`DivPart`

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest)
data(Paracou618)
# Estimate diversity.
Profile <- DivProfile(q.seq = seq(0, 2, 0.1), Paracou618.MC, Biased = FALSE)
plot(Profile)
autoplot(Profile)
summary(Profile)
```

Dqz *Similarity-based diversity of a community*

Description

Calculates the diversity of order q of a probability vector according to a similarity matrix.

Usage

```
Dqz(NorP, q = 1, Z = diag(length(NorP)), ...)
bcDqz(Ns, q = 1, Z = diag(length(Ns)), Correction = "Best", CheckArguments = TRUE)
## S3 method for class 'ProbaVector'
Dqz(NorP, q = 1, Z = diag(length(NorP)), ...,
  CheckArguments = TRUE, Ps = NULL)
## S3 method for class 'AbdVector'
Dqz(NorP, q = 1, Z = diag(length(NorP)), Correction = "Best", ...,
  CheckArguments = TRUE, Ns = NULL)
## S3 method for class 'integer'
Dqz(NorP, q = 1, Z = diag(length(NorP)), Correction = "Best", ...,
  CheckArguments = TRUE, Ns = NULL)
## S3 method for class 'numeric'
Dqz(NorP, q = 1, Z = diag(length(NorP)), Correction = "Best", ...,
  CheckArguments = TRUE, Ps = NULL, Ns = NULL)
```

Arguments

`Ps`	A probability vector, summing to 1.
`Ns`	A numeric vector containing species abundances.
`NorP`	A numeric vector, an integer vector, an abundance vector (`AbdVector`) or a probability vector (`ProbaVector`). Contains either abundances or probabilities.
`q`	A number: the order of diversity. Default is 1.
`Z`	A relatedness matrix, *i.e.* a square matrix whose terms are all positive, strictly positive on the diagonal. Generally, the matrix is a similarity matrix, *i.e.* the diagonal terms equal 1 and other terms are between 0 and 1. Default is the identity matrix to calculate neutral diversity.
`Correction`	A string containing one of the possible corrections: `"None"` (no correction), `"HorvitzThomson"`, `"MarconZhang"` or `"Best"`, the default value. The `"MarconZhang"` correction assumes a similarity matrix.
`...`	Additional arguments. Unused.
`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

Diversity is calculated following Leinster and Cobbold (2012): it is the reciprocal of the (generalized) average (of order `q`) of the community species ordinariness.

A similarity matrix is used (as for `Dqz`), not a distance matrix as in Ricotta and Szeidl (2006). See the example.

Bias correction requires the number of individuals. Use `bcHqz` and choose the `Correction`. Correction techniques are from Marcon *et al.* (2014).

Currently, the `"Best"` correction is the max value of `"HorvitzThomson"` and `"MarconZhang"`.

The functions are designed to be used as simply as possible. `Dqz` is a generic method. If its first argument is an abundance vector, an integer vector or a numeric vector which does not sum to

1, the bias corrected function `bcDqz` is called. Explicit calls to `bcDqz` (with bias correction) or to `Dqz.ProbaVector` (without correction) are possible to avoid ambiguity. The `.integer` and `.numeric` methods accept `Ps` or `Ns` arguments instead of `NorP` for backward compatibility.

Value

A named number equal to the calculated diversity. The name is that of the bias correction used.

References

Leinster, T. and Cobbold, C. (2012). Measuring diversity: the importance of species similarity. *Ecology* 93(3): 477-489.

Marcon, E., Zhang, Z. and Herault, B. (2014). The decomposition of similarity-based diversity and its bias correction. *HAL* hal-00989454(version 3).

See Also

`Hqz`, `PhyloDiversity`

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest)
data(Paracou618)
# Prepare the similarity matrix
DistanceMatrix <- as.matrix(Paracou618.dist)
# Similarity can be 1 minus normalized distances between species
Z <- 1 - DistanceMatrix/max(DistanceMatrix)
# Calculate diversity of order 2
Dqz(Paracou618.MC$Ns, 2, Z)
```

`EightSpAbundance` *Abundances of 8 species to run examples.*

Description

This dataset is a light-weight example.

Usage

```
data(Paracou618)
```

Format

A named vector.

Examples

```
data(Paracou618)
EightSpAbundance
```

`EightSpTree` *Functional tree with 8 species.*

Description

This dataset is a leight-weight example.

Usage

```
data(Paracou618)
```

Format

An object of class `phylog` containing a functional tree.

Examples

```
data(Paracou618)
# Preprocess the tree to be able to plot it
# without loading ade4 package
plot(Preprocess.Tree(EightSpTree), hang=-0.01)
```

`Enq` *Grassberger's expectation of n^q*

Description

Expected value of n^q when n follows a Poisson law.

Usage

```
Enq(n, q)
```

Arguments

`n`	A positive integer vector.
`q`	A positive number.

Details

The expectation of n^q when n follows a Poisson ditribution has been derived by Grassberger (1988).

Value

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

Note

The function is computed using the `beta`.function.

Its value is 0 for $n - q + 1 < 0$.

References

Grassberger, P. (1988). Finite sample corrections to entropy and dimension estimates. Physics Letters A 128(6-7): 369-373.

Examples

```
# Compare
n <- c(2,3)
Enq(n, q=2)
# with
n^2

# Result is 1
Enq(n, q=0)
# Result is 0
Enq(n, q=5)
```

`EntropyCI` *Entropy of Monte-Carlo simulated communities*

Description

Resamples a community by Monte-Carlo simulations of a multinomial distribution and returns a vector of entropy values to calculate confidence intervals.

Usage

```
EntropyCI(FUN, Simulations = 100, Ns, BootstrapMethod = "Chao2015",
    ShowProgressBar = TRUE, ..., CheckArguments = TRUE)
```

Arguments

`FUN`	The entropy function to be applied to each simulated community. May be any entropy function accepting a vector of species abundances, such as `bcTsallis`, `bcShannon`, `bcSimpson` or `bcPhyloEntropy`.
`Simulations`	The number of simulations to build confidence intervals.
`Ns`	A numeric vector containing species abundances.
`BootstrapMethod`	The method used to obtain the probabilities to generate bootstrapped communities from observed abundances. See `rCommunity`.
`...`	Additional arguments to be passed to `FUN`.

`ShowProgressBar`	If `TRUE` (default), a progress bar is shown.
`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

This function is used to obtain the distribution of entropy and eventually calculate confidence intervals. It draws simulated communities according to a multinomial distribution with the same number of individuals and probabilities as the actual community. It calculates the entropy of each simulated community. Last, it recenters the distribution of entropy values arounf the actual value of entropy according to Marcon *et al.* (2012): the estimation bias of simulated communities entropy can not be corrected analytically, but it does not affect the distribution shape.

Diversity can not be recentered this way so diversity function should not be used. Unexpected results will be obtained if inappropriate functions are used.

Value

A numeric vector containing the entropy value of each simulated community.

References

Marcon, E., Herault, B., Baraloto, C. and Lang, G. (2012). The Decomposition of Shannon's Entropy and a Confidence Interval for Beta Diversity. *Oikos* 121(4): 516-522.

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest)
data(Paracou618)
# Abundance (all estimators will include bias corrrection)
Ns <- as.AbdVector(Paracou618.MC$Ns)
q <- 1
# Estimate entropy and transform it into diversity
RealEst <- expq(Tsallis(Ns, q), q)
# Transform the distribution of Tsallis entropy into diversity
SimulatedDiversity <- expq(EntropyCI(Tsallis, Simulations=50, Ns, q=q), q)
# Figure
plot(density(SimulatedDiversity), col="black", lwd=2, main="", xlab ="Diversity")
abline(v=RealEst, col="red", lwd=2, lty=2)
abline(v=quantile(SimulatedDiversity, probs = 0.025), col="black", lwd=1, lty=3)
abline(v=quantile(SimulatedDiversity, probs = 0.975), col="black", lwd=1, lty=3)
legend("topright", c("Real value", "Confidence interval"), lty=c(2,3),
col=c("red", "black"), inset=0.01)
# Print results
cat("Estimated Diversity:", RealEst)
quantile(SimulatedDiversity, probs = c(0.025, 0.975))
```

`expq` *Exponential of order q*

Description

Calculates the deformed exponential of order q.

Usage

```
expq(x, q)
expq.CommunityProfile(Profile)
```

Arguments

`x`	A numeric vector.
`Profile`	A `CommunityProfile`.
`q`	A number.

Details

The deformed exponential 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 $x > \frac{1}{(q-1)}$.

`expq.CommunityProfile` calculates the deformed exponential of a `CommunityProfile`. Its `$x` item (the order of diversity) is kept unchanged whilst other items are set to their exponential of order `$x`. Thus, an entropy profile is transformed into a diversity profile.

Value

A vector of the same length as `x` containing the transformed values or a `CommunityProfile`.

References

Marcon, E., Scotti, I., Herault, B., Rossi, V. and Lang, G. (2014). Generalization of the partitioning of Shannon diversity. *PLOS One* 9(3): e90289.

Tsallis, C. (1994). What are the numbers that experiments provide? *Quimica Nova* 17(6): 468-471.

See Also

`expq`

Examples

```
curve(exp(x), -5, 0, lty=3)
curve(expq(x, 2), -5, 0, lty=2, add=TRUE)
curve(expq(x, 3), -5, 0, lty=1, add=TRUE)
legend("topleft", legend = c("exp(x)", "exp2(x)", "exp3(x)"), lty = c(1, 2, 3), inset=0.02)
```

`GammaDiversity` *Reduced-bias gamma diversity of a metacommunity*

Description

Calculates the reduced-bias diversity of order q of a metacommunity.

Usage

```
GammaDiversity(MC, q = 1, Correction = "Best", Tree = NULL, Normalize = TRUE,
  Z = NULL, CheckArguments = TRUE)
```

Arguments

`MC`	A `MetaCommunity` object.
`q`	A number: the order of diversity. Default is 1.
`Correction`	A string containing one of the possible corrections accepted by `AlphaEntropy` or `"None"` or `"Best"`, the default value.
`Tree`	An object of class `hclust`, "phylo" (see `read.tree`), `phylog` or `PPtree`. The tree must be ultrametric.
`Normalize`	If `TRUE` (default), diversity is not affected by the height of the tree. If `FALSE`, diversity is proportional to the height of the tree.
`Z`	A relatedness matrix, *i.e.* a square matrix whose terms are all positive, strictly positive on the diagonal. Generally, the matrix is a similarity matrix, *i.e.* the diagonal terms equal 1 and other terms are between 0 and 1.
`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

Entropy is calculated by `GammaEntropy` and transformed into diversity.

Value

The metacommunity's gamma entropy.

References

Marcon, E., Scotti, I., Herault, B., Rossi, V. and Lang, G. (2014). Generalization of the partitioning of Shannon diversity. *PLOS One* 9(3): e90289.

Marcon, E., Herault, B. (2015). Decomposing Phylodiversity. *Methods in Ecology and Evolution* 6(3): 333-339.

Marcon, E., Zhang, Z. and Herault, B. (2014). The decomposition of similarity-based diversity and its bias correction. *HAL* hal-00989454(version 3).

See Also

`GammaEntropy`

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest)
data(Paracou618)
# Calculate Simpson gamma diversity
GammaDiversity(Paracou618.MC, 2)
# Compare without correction
GammaDiversity(Paracou618.MC, 2, Correction = "None")
# Estimate phylogenetic Simpson gamma diversity
GammaDiversity(Paracou618.MC, 2, Tree = Paracou618.Taxonomy)
```

`GammaEntropy` *Reduced-bias gamma entropy of a metacommunity*

Description

Calculates the reduced-bias Tsallis entropy of order q of a metacommunity.

Usage

```
GammaEntropy(MC, q = 1, Correction = "Best", Tree = NULL, Normalize = TRUE,
  Z = NULL, PhyloDetails = FALSE, CheckArguments = TRUE)
```

Arguments

`MC`	A `MetaCommunity` object.
`q`	A number: the order of entropy. Default is 1.
`Correction`	A string containing one of the possible corrections accepted by the bias-corrected entropy function (see details) or `"None"` or `"Best"`, the default value.
`Tree`	An object of class `hclust`, "phylo" (see `read.tree`), `phylog` or `PPtree`. The tree must be ultrametric.
`Normalize`	If `TRUE` (default), the entropy returned by the function is normalized by the height of the tree (it is the weighted average value of the entropy in each slice). If `FALSE`, it is the unnormalized weighted sum of the results.
`Z`	A relatedness matrix, *i.e.* a square matrix whose terms are all positive, strictly positive on the diagonal. Generally, the matrix is a similarity matrix, *i.e.* the diagonal terms equal 1 and other terms are between 0 and 1.
`PhyloDetails`	If `FALSE` (default), the function always returns a number. If `TRUE` and `Tree` is not Tree is not `NULL` then a `PhyloValue` object is returned with all details. That is used internally by `DivPart` to obtain the corrections used to estimate gamma entropy along the tree and apply them to the estimation of alpha diversity.
`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

If `Tree` is not `NULL`, then phylogenetic entropy is calculated by `bcPhyloEntropy`.

Else, if `Z` is not `NULL`, then similarity-based entropy is calculated by `bcHqz`.

Else, neutral entropy is calculated by `bcTsallis`.

Value

A number equal to the calculated entropy.

References

Marcon, E., Scotti, I., Herault, B., Rossi, V. and Lang, G. (2014). Generalization of the partitioning of Shannon diversity. *PLOS One* 9(3): e90289.

Marcon, E., Herault, B. (2015). Decomposing Phylodiversity. *Methods in Ecology and Evolution* 6(3): 333-339.

Marcon, E., Zhang, Z. and Herault, B. (2014). The decomposition of similarity-based diversity and its bias correction. *HAL* hal-00989454(version 3).

See Also

`bcTsallis`, `bcPhyloEntropy`

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest)
data(Paracou618)
# Calculate Simpson gamma entropy
GammaEntropy(Paracou618.MC, 2)
# Compare without correction
GammaEntropy(Paracou618.MC, 2, Correction = "None")
# Estimate phylogenetic Simpson gamma entropy
GammaEntropy(Paracou618.MC, 2, Tree = Paracou618.Taxonomy)
```

`GenSimpson` *Generalized Simpson's Entropy and Diversity*

Description

Calculates the Generalized Simpson's entropy of order r of a probability or abundance vector, and its effective number of species.

Usage

```
GenSimpson(NorP, r = 1, ...)
bcGenSimpson(Ns, r = 1, CheckArguments = TRUE)
## S3 method for class 'ProbaVector'
GenSimpson(NorP, r = 1, ...,
  CheckArguments = TRUE, Ps = NULL)
## S3 method for class 'AbdVector'
GenSimpson(NorP, r = 1, ...,
  CheckArguments = TRUE, Ns = NULL)
## S3 method for class 'integer'
GenSimpson(NorP, r = 1, ...,
  CheckArguments = TRUE, Ns = NULL)
## S3 method for class 'numeric'
GenSimpson(NorP, r = 1, ...,
  CheckArguments = TRUE, Ps = NULL, Ns = NULL)
GenSimpsonD(NorP, r = 1, ...)
bcGenSimpsonD(Ns, r = 1, CheckArguments = TRUE)
## S3 method for class 'ProbaVector'
GenSimpsonD(NorP, r = 1, ...,
  CheckArguments = TRUE, Ps = NULL)
## S3 method for class 'AbdVector'
GenSimpsonD(NorP, r = 1, ...,
  CheckArguments = TRUE, Ns = NULL)
## S3 method for class 'integer'
GenSimpsonD(NorP, r = 1, ...,
  CheckArguments = TRUE, Ns = NULL)
## S3 method for class 'numeric'
GenSimpsonD(NorP, r = 1, ...,
  CheckArguments = TRUE, Ps = NULL, Ns = NULL)
```

Arguments

`Ps`	A probability vector, summing to 1.
`Ns`	A numeric vector containing species abundances.
`NorP`	A numeric vector, an integer vector, an abundance vector (`AbdVector`) or a probability vector (`ProbaVector`). Contains either abundances or probabilities.
`r`	A number: the order of diversity. Default is 1 for Simpson's diversity.
`...`	Additional arguments. Unused.
`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

The Generalized Simpson's Entropy (Zhang and Zhou, 2010) of order r is, in the species accumulation curve, the probability for the individual sampled in rank $r + 1$ to belong to a new species. It is a measure of diversity so long as r is lower than the number of species (Grabchak *et al.*, 2016).

Bias correction requires the number of individuals. Use `bcGenSimpson`. It is limited to orders r less than or equal to the number of individuals in the community.

The effective number of species `GenSimpsonD` (explicit diversity) has been derived by Grabchak *et al.* (2016).

The functions are designed to be used as simply as possible. `GenSimpson` is a generic method. If its first argument is an abundance vector, an integer vector or a numeric vector which does not sum to 1, the bias corrected function `bcGenSimpson` is called. Explicit calls to `bcGenSimpson` (with bias correction) or to `GenSimpson.ProbaVector` (without correction) are possible to avoid ambiguity. The `.integer` and `.numeric` methods accept `Ps` or `Ns` arguments instead of `NorP` for backward compatibility.

Value

A named number equal to the calculated index or diversity. The name is either "Biased" or "Unbiased", depending on the estimator used.

Note

The unbiased estimator is calculated by the `GenSimp.z` function of the `EntropyEstimation` package.

References

Grabchak, M., Marcon, E., Lang, G., and Zhang, Z. (2017). The Generalized Simpson's Entropy is a Measure of Biodiversity. *Plos One*, 12(3): e0173305.

Zhang Z. and Zhou J. (2010). Re-parameterization of multinomial distributions and diversity indices. *Journal of Statistical Planning and Inference* 140(7): 1731-1738.

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest)
data(Paracou618)
# Ns is the total number of trees per species
Ns <- as.AbdVector(Paracou618.MC$Ns)
# Species probabilities
Ps <- as.ProbaVector(Paracou618.MC$Ns)
# Whittaker plot
plot(Ns)

# Calculate GenSimpson entropy of order 1, equal to Simpson's index of diversity
GenSimpson(Ps, 1)
# Calculate an unbiased estimator of GenSimpson diversity of order 100
GenSimpsonD(Ns, 100)
```

Hqz *Similarity-based entropy of a community*

Description

Calculates the entropy of order q of a probability vector according to a similarity matrix.

Usage

```
Hqz(NorP, q = 1, Z = diag(length(NorP)),  ...)
bcHqz(Ns, q = 1, Z = diag(length(Ns)), Correction = "Best", SampleCoverage = NULL,
  CheckArguments = TRUE)
## S3 method for class 'ProbaVector'
Hqz(NorP, q = 1, Z = diag(length(NorP)),
  ..., CheckArguments = TRUE, Ps = NULL)
## S3 method for class 'AbdVector'
Hqz(NorP, q = 1, Z = diag(length(NorP)), Correction = "Best",
  ..., CheckArguments = TRUE, Ns = NULL)
## S3 method for class 'integer'
Hqz(NorP, q = 1, Z = diag(length(NorP)), Correction = "Best",
  ..., CheckArguments = TRUE, Ns = NULL)
## S3 method for class 'numeric'
Hqz(NorP, q = 1, Z = diag(length(NorP)), Correction = "Best",
  ..., CheckArguments = TRUE, Ps = NULL, Ns = NULL)
```

Arguments

`Ps`	A probability vector, summing to 1.
`Ns`	A numeric vector containing species abundances.
`NorP`	A numeric vector, an integer vector, an abundance vector (`AbdVector`) or a probability vector (`ProbaVector`). Contains either abundances or probabilities.
`q`	A number: the order of entropy. Default is 1.
`Z`	A relatedness matrix, *i.e.* a square matrix whose terms are all positive, strictly positive on the diagonal. Generally, the matrix is a similarity matrix, *i.e.* the diagonal terms equal 1 and other terms are between 0 and 1. Default is the identity matrix to calculate neutral entropy.
`Correction`	A string containing one of the possible corrections: `"None"` (no correction), `"ChaoShen"`, `"MarconZhang"` or `"Best"`, the default value. The `"MarconZhang"` correction assumes a similarity matrix.
`SampleCoverage`	The sample coverage of `Ns` calculated elsewhere. Used to calculate the gamma diversity of meta-communities, see details.
`...`	Additional arguments. Unused.
`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

Entropy is calculated following Leinster and Cobbold (2012) after Ricotta and Szeidl (2006): it is the entropy of order `q` of the community, using species ordinariness as the information function.

A similarity matrix is used (as for `Dqz`), not a distance matrix as in Ricotta and Szeidl (2006). See the example.

Bias correction requires the number of individuals. Use `bcHqz` and choose the `Correction`. Correction techniques are from Marcon *et al.* (2014).

Currently, the `"Best"` correction is the max value of `"ChaoShen"` and `"MarconZhang"`.

The functions are designed to be used as simply as possible. `Hqz` is a generic method. If its first argument is an abundance vector, an integer vector or a numeric vector which does not sum to 1, the bias corrected function `bcHqz` is called. Explicit calls to `bcHqz` (with bias correction) or to `Hqz.ProbaVector` (without correction) are possible to avoid ambiguity. The `.integer` and `.numeric` methods accept `Ps` or `Ns` arguments instead of `NorP` for backward compatibility.

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 `SampleCoverage` argument allows applying the `"ChaoShen"` correction to estimate quite well the entropy. `DivPart` and `GammaEntropy` functions use this tweak.

Value

A named number equal to the calculated entropy. The name is that of the bias correction used.

References

Leinster, T. and Cobbold, C. (2012). Measuring diversity: the importance of species similarity. *Ecology* 93(3): 477-489.

Marcon, E., Zhang, Z. and Herault, B. (2014). The decomposition of similarity-based diversity and its bias correction. *HAL* hal-00989454(version 3).

Ricotta, C. and Szeidl, L. (2006). Towards a unifying approach to diversity measures: Bridging the gap between the Shannon entropy and Rao's quadratic index. *Theoretical Population Biology* 70(3): 237-243.

See Also

`Dqz`, `PhyloEntropy`

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest)
data(Paracou618)
# Prepare the similarity matrix
DistanceMatrix <- as.matrix(EightSpTree$Wdist^2/2)
# Similarity can be 1 minus normalized distances between species
Z <- 1 - DistanceMatrix/max(DistanceMatrix)
# Calculate diversity of order 2
Ps <- EightSpAbundance/sum(EightSpAbundance)
Hqz(Ps, 2, Z)
```

```
# Equal to normalized Rao quadratic entropy when q=2
Rao(Ps, EightSpTree)/max(DistanceMatrix)
# But different from PhyloEntropy for all other q, e.g. 1
Hqz(Ps, 1, Z)
summary(PhyloEntropy(Ps, 1, EightSpTree))
```

`HqzBeta` *Similarity-based beta entropy of a community*

Description

Calculates the similarity-based beta entropy of order q of a community belonging to a metacommunity.

Usage

```
HqzBeta(NorP, NorPexp = NULL, q = 1, Z = diag(length(NorP)), ...)
bcHqzBeta(Ns, Nexp = NULL, q = 1, Z = diag(length(Ns)), Correction = "Best",
          CheckArguments = TRUE)
## S3 method for class 'ProbaVector'
HqzBeta(NorP, NorPexp = NULL, q = 1, Z = diag(length(NorP)),
  ..., CheckArguments = TRUE, Ps = NULL, Pexp = NULL)
## S3 method for class 'AbdVector'
HqzBeta(NorP, NorPexp = NULL, q = 1, Z = diag(length(NorP)), Correction = "Best",
  ..., CheckArguments = TRUE, Ns = NULL, Nexp = NULL)
## S3 method for class 'integer'
HqzBeta(NorP, NorPexp = NULL, q = 1, Z = diag(length(NorP)), Correction = "Best",
  ..., CheckArguments = TRUE, Ns = NULL, Nexp = NULL)
## S3 method for class 'numeric'
HqzBeta(NorP, NorPexp = NULL, q = 1, Z = diag(length(NorP)), Correction = "Best",
  ..., CheckArguments = TRUE, Ps = NULL, Ns = NULL, Pexp = NULL, Nexp = NULL)
```

Arguments

`Ps`	The probability vector of species of the community.
`Pexp`	The probability vector of species of the metacommunity.
`Ns`	A numeric vector containing species abundances of the community.
`Nexp`	A numeric vector containing species abundances of the metacommunity.
`NorP`	A numeric vector, an integer vector, an abundance vector (`AbdVector`) or a probability vector (`ProbaVector`). Contains either abundances or probabilities of the community.
`NorPexp`	A numeric vector, an integer vector, an abundance vector (`AbdVector`) or a probability vector (`ProbaVector`). Contains either abundances or probabilities of the metacommunity.
`q`	A number, the order of diversity. Default is 1.

`Z`	A relatedness matrix, *i.e.* a square matrix whose terms are all positive, strictly positive on the diagonal. Generally, the matrix is a similarity matrix, *i.e.* the diagonal terms equal 1 and other terms are between 0 and 1. Default is the identity matrix to calculate neutral entropy.
`Correction`	A string containing one of the possible corrections: currently, no correction is available so `"Best"`, the default value, is equivalent to `"None"`.
`...`	Additional arguments. Unused.
`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

The derivation of similarity-based beta entropy can be found in Marcon *et al.* (2014).

Bias correction requires the number of individuals.

Note that beta entropy value is related to alpha entropy (if q is not 1) and cannot be compared accross communities (Jost, 2007). Beta entropy of a community is not meaningful in general, do rather calculate the `BetaDiversity` of the metacommunity.

The functions are designed to be used as simply as possible. `HqzBeta` is a generic method. If its first argument is an abundance vector, an integer vector or a numeric vector which does not sum to 1, the bias corrected function `bcHqzBeta` is called. Explicit calls to `bcHqzBeta` (with bias correction) or to `HqzBeta.ProbaVector` (without correction) are possible to avoid ambiguity. The `.integer` and `.numeric` methods accept `Ps` or `Ns` arguments instead of `NorP` for backward compatibility.

Value

A named number equal to the calculated entropy. The name is that of the bias correction used.

References

Jost (2007), Partitioning diversity into independent alpha and beta components. *Ecology* 88(10): 2427-2439.

Marcon, E., Zhang, Z. and Herault, B. (2014). The decomposition of similarity-based diversity and its bias correction. *HAL* hal-00989454(version 3).

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest)
data(Paracou618)
# Ps is the vector of probabilities
Ps <- as.ProbaVector(Paracou618.MC$Ps)
# Probability distribution of the first plot
Ps1 <- as.ProbaVector(Paracou618.MC$Psi[, 1])
# Prepare the similarity matrix
DistanceMatrix <- as.matrix(Paracou618.dist)
# Similarity can be 1 minus normalized distances between species
Z <- 1 - DistanceMatrix/max(DistanceMatrix)
# Divergence of order 2 between plot 1 and the whole forest
HqzBeta(Ps1, Ps, q=2, Z)
```

`Hurlbert` *Hurlbert's Index and Explicit Diversity*

Description

Calculates the Hurlbert entropy of order k of a probability or abundance vector, and its effective number of species.

Usage

```
Hurlbert(NorP, k = 2, ...)
bcHurlbert(Ns, k = 2, CheckArguments = TRUE)
## S3 method for class 'ProbaVector'
Hurlbert(NorP, k = 2, ...,
  CheckArguments = TRUE, Ps = NULL)
## S3 method for class 'AbdVector'
Hurlbert(NorP, k = 2, ...,
  CheckArguments = TRUE, Ns = NULL)
## S3 method for class 'integer'
Hurlbert(NorP, k = 2, ...,
  CheckArguments = TRUE, Ns = NULL)
## S3 method for class 'numeric'
Hurlbert(NorP, k = 2, ...,
  CheckArguments = TRUE, Ps = NULL, Ns = NULL)
HurlbertD(NorP, k = 2, ...)
bcHurlbertD(Ns, k = 2, CheckArguments = TRUE)
## S3 method for class 'ProbaVector'
HurlbertD(NorP, k = 2, ...,
  CheckArguments = TRUE, Ps = NULL)
## S3 method for class 'AbdVector'
HurlbertD(NorP, k = 2, ...,
  CheckArguments = TRUE, Ns = NULL)
## S3 method for class 'integer'
HurlbertD(NorP, k = 2, ...,
  CheckArguments = TRUE, Ns = NULL)
## S3 method for class 'numeric'
HurlbertD(NorP, k = 2, ...,
  CheckArguments = TRUE, Ps = NULL, Ns = NULL)
```

Arguments

`Ps`	A probability vector, summing to 1.
`Ns`	A numeric vector containing species abundances.
`NorP`	A numeric vector, an integer vector, an abundance vector (`AbdVector`) or a probability vector (`ProbaVector`). Contains either abundances or probabilities.
`k`	A number: the order of diversity. Default is 2 for Simpson's diversity.

`...`	Additional arguments. Unused.
`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

Hurlbert's index of diversity (1971) of order k is the expected number of species in a sample of size k.

Bias correction requires the number of individuals. Use `bcHurlbert`. It is limited to orders k less than or equal to the number of individuals in the community.

The effective number of species `HurlbertD` (explicit diversity) has been derived by Dauby & Hardy (2012). It is calculated numerically. `bcHurlbertD` calculates it from the bias-corrected index `bcHurlbert`.

The functions are designed to be used as simply as possible. `Hurlbert` is a generic method. If its first argument is an abundance vector, an integer vector or a numeric vector which does not sum to 1, the bias corrected function `bcHurlbert` is called. Explicit calls to `bcHurlbert` (with bias correction) or to `Hurlbert.ProbaVector` (without correction) are possible to avoid ambiguity. The `.integer` and `.numeric` methods accept `Ps` or `Ns` arguments instead of `NorP` for backward compatibility.

Value

A named number equal to the calculated index or diversity. The name is either "Biased" or "Unbiased", depending on the estimator used.

References

Dauby G. & Hardy O.J. (2012) Sampled-based estimation of diversity sensu stricto by transforming Hurlbert diversities into effective number of species. *Ecography* 35(7): 661-672.

Hurlbert (1971) The Nonconcept of Species Diversity: A Critique and Alternative Parameters. *Ecology* 52(4): 577-586.

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest)
data(Paracou618)
# Ns is the total number of trees per species
Ns <- as.AbdVector(Paracou618.MC$Ns)
# Species probabilities
Ps <- as.ProbaVector(Paracou618.MC$Ns)
# Whittaker plot
plot(Ns)

# Calculate Hurlbert entropy of order 2, equal to Simpson's index of diversity
Hurlbert(Ps, 2)
# Calculate an unbiased estimator of Hurlbert entropy of order 2
Hurlbert(Ns, 2)
```

`KLq` *Generalized Kullback-Leibler divergence*

Description

Calculates the generalized Kullback-Leibler divergence between an observed and an expected probability distribution.

Usage

```
KLq(Ps, Pexp, q = 1, CheckArguments = TRUE)
```

Arguments

`Ps`	The observed probability vector.
`Pexp`	The expected probability vector.
`q`	A number: the order of entropy. Default is 1.
`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

The generalized Kullback-Leibler divergence (Borland et al., 1998) converges to the Kullback-Leibler divergence (Kullback and Leibler, 1951) when q tends to 1. It is used to calculate the generalized beta entropy (Marcon et al., 2014).

Value

A number equal to the generalized Kullback-Leibler divergence between the probability distributions.

References

Borland, L., Plastino, A. R. and Tsallis, C. (1998). Information gain within nonextensive thermostatistics. *Journal of Mathematical Physics* 39(12): 6490-6501.

Kullback, S. and Leibler, R. A. (1951). On Information and Sufficiency. The Annals of Mathematical *Statistics* 22(1): 79-86.

Marcon, E., Scotti, I., Herault, B., Rossi, V. and Lang, G. (2014). Generalization of the partitioning of Shannon diversity. *PLOS One* 9(3): e90289.

See Also

`TsallisBeta`

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest)
data(Paracou618)
# Ps is the vector of probabilities
Ps <- Paracou618.MC$Ps
# Probability distribution of the first plot
Ps1 <- Paracou618.MC$Psi[, 1]
# Divergence of order 2 between the first plot and the whole forest
KLq(Ps1, Ps, 2)
```

`lnq` *Logarithm of order q*

Description

Calculates the deformed logarithm of order q.

Usage

```
lnq(x, q)
lnq.CommunityProfile(Profile)
```

Arguments

`x`	A numeric vector.
`Profile`	A `CommunityProfile`.
`q`	A number.

Details

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

`lnq.CommunityProfile` calculates the deformed logarithm of a `CommunityProfile`. Its `$x` item (the order of diversity) is kept unchanged whilst other items are set to their logarithm of order `$x`. Thus, a diversity profile is transformed into an entropy profile.

Value

A vector of the same length as `x` containing the transformed values or a `CommunityProfile`.

References

Tsallis, C. (1994). What are the numbers that experiments provide? *Quimica Nova* 17(6): 468-471.

See Also

`expq`

Examples

```
curve(log(x), 0, 1, lty=1)
curve(lnq(x, 2), 0, 1, lty=2, add=TRUE)
curve(lnq(x, 3), 0, 1, lty=3, add=TRUE)
legend("topleft", legend = c("log(x)", "ln2(x)", "ln3(x)"), lty = c(1, 2, 3), inset=0.02)
```

`MC Utilities` *Manipulation of meta-communities*

Description

Tools to manipulate meta-communities. From a list of meta-communities, `MergeMC` creates a meta-community whose communities are each original metacommunity. `MergeC` creates a metacommunity whose communities are each original community. `ShuffleMC` randomly assigns original communities to a metacommunity, keeping original weights, and returns a list of meta-communities.

Usage

```
MergeMC(MClist, Weights = rep(1, length(MClist)), CheckArguments = TRUE)
MergeC(MClist, Weights = rep(1, length(MClist)), CheckArguments = TRUE)
ShuffleMC(MClist, Weights = rep(1, length(MClist)), CheckArguments = TRUE)
```

Arguments

`MClist`	A list of `MetaCommunity` objects.
`Weights`	A vector of numbers containing the weight of each metacommunity of the list. It does not have to be normalized to sum to 1.
`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

`MergeMC` is used for hierarchical partitioning of diversity. The gamma diversity of communities of the list becomes alpha diversity of the merged meta-community.

`MergeC` creates a new meta-community by mixing original ones. Original communities are kept, their new weight is the product of their original weight and the weight of their original meta-community.

`ShuffleMC` is used for simulations of the null hypothesis that all metacommunities of the list are identical.

Value

`MergeMC` and `MergeC` return a `MetaCommunity`.

`ShuffleMC` returns a `list` of `MetaCommunity` objects.

See Also

`MetaCommunity`

Examples

```
# First meta-community
(df <- data.frame(C1 = c(10, 10, 10, 10), C2 = c(0, 20, 35, 5),
  C3 = c(25, 15, 0, 2), row.names = c("sp1", "sp2", "sp3", "sp4")))
w <- c(1, 2, 1)
MC1 <- MetaCommunity(Abundances = df, Weights = w)
# Second meta-community
(df <- data.frame(C1 = c(10, 4), C2 = c(3, 4), row.names = c("sp1", "sp5")))
w <- c(3, 2)
MC2 <- MetaCommunity(Abundances = df, Weights = w)

# Merge communities
plot(MergeC(list(MC1, MC2)), main="Merged communities")
# Merge metacommunities
plot(MergeMC(list(MC1, MC2)), main="Merged meta-communities")
smc <- ShuffleMC(list(MC1, MC2))
plot(MergeMC(smc), main="Shuffled, then Merged meta-communities")
```

`MCdiversity` *Meta-Community diversity class.*

Description

Methods for objects of type "`MCdiversity`".

Usage

```
is.MCdiversity(x)
## S3 method for class 'MCdiversity'
plot(x, ...)
## S3 method for class 'MCdiversity'
autoplot(object, col = "grey35",
          border = NA, ...)
## S3 method for class 'MCdiversity'
summary(object, ...)
```

Arguments

`x`	An object to be tested or plotted.
`object`	A `MCdiversity` object to be summarized or plotted.
`col`	The color used to fill the bars. See "Color Specification" in `par`.
`border`	The color of the borders around the bars. See `hist`.
`...`	Additional arguments to be passed to the generic methods.

Value

Meta-community diversity objects are lists containing:

`MetaCommunity`	The name of the `MetaCommunity` object containing inventory data.
`Type`	The type of diversity ("alpha", "beta" or "gamma").
`Order`	The order of diversity q.
`Correction`	The estimation bias correction used to calculate diversity.
`Tree`	The phylogenetic or functional tree used to calculate phylodiversity.
`Normalized`	Logical. Indicates whether phylodiversity is normalized or proportional to the height of the tree.
`Weights`	A vector containing the weights of communities.
`Communities`	A vector containing the diversity of communities.
`Total`	The total diversity.

`is.MCdiversity` returns `TRUE` if the object is of class `MCdiversity`.

`summary.MCdiversity` returns a summary of the object's value.

`MCentropy` *Meta-Community entropy class.*

Description

Methods for objects of type "`MCentropy`".

Usage

```
is.MCentropy(x)
## S3 method for class 'MCentropy'
plot(x, ...)
## S3 method for class 'MCentropy'
autoplot(object, col = "grey35",
          border = NA, ...)
## S3 method for class 'MCentropy'
summary(object, ...)
```

Arguments

`x`	An object to be tested or plotted.
`object`	A `MCentropy` object to be summarized or plotted.
`col`	The color used to fill the bars. See "Color Specification" in `par`.
`border`	The color of the borders around the bars. See `hist`.
`...`	Additional arguments to be passed to the generic methods.

Value

Meta-community entropy objects are lists containing:

`MetaCommunity`	The name of the `MetaCommunity` object containing inventory data.
`Method`	The method used to calculate entropy ("HCDT", "Similarity-based").
`Type`	The type of entropy ("alpha", "beta" or "gamma").
`Order`	The order of entropy q.
`Correction`	The estimation bias correction used to calculate entropy.
`Tree`	The phylogenetic or functional tree used to calculate phyloentropy.
`Normalized`	Logical. Indicates whether phyloentropy is normalized or proportional to the height of the tree.
`Z`	The matrix used to calculate similarity-based entropy.
`Weights`	A vector containing the weights of communities.
`Communities`	A vector containing the entropy of communities.
`Total`	The total entropy.

`is.MCentropy` returns `TRUE` if the object is of class `MCentropy`.

`summary.MCentropy` returns a summary of the object's value.

`MetaCommunity` *Metacommunity class*

Description

Methods for objects of type "`MetaCommunity`".

Usage

```
MetaCommunity(Abundances, Weights = rep(1, ncol(Abundances)))
is.MetaCommunity(x)
## S3 method for class 'MetaCommunity'
summary(object, ...)
## S3 method for class 'MetaCommunity'
plot(x, ...)
```

Arguments

`Abundances`	A dataframe containing the number of observations (lines are species, columns are communities). The first column of the dataframe may contain the species names.
`Weights`	A vector of positive numbers equal to community weights or a dataframe containing a vector named `Weights`. It does not have to be normalized. Weights are equal by default.
`x`	An object to be tested or plotted.
`object`	A `MetaCommunity` object to be summarized.
`...`	Additional arguments to be passed to the generic methods.

Details

In the entropart package, individuals of different "species" are counted in several "communities" which are agregated to define a "metacommunity".

This is a naming convention, which may correspond to plots in a forest inventory or any data organized the same way.

Alpha and beta entropies of communities are summed according to `Weights` and the probability to find a species in the metacommunity is the weighted average of probabilities in communities.

The simplest way to import data is to organize it into two text files. The first file should contain abundance data: the first column named `Species` for species names, and a column for each community.

The second file should contain the community weights in two columns. The first one, named `Communities` should contain their names and the second one, named `Weights`, their weights.

Files can be read and data imported by code such as:

```
Abundances <- read.csv(file="Abundances.csv", row.names = 1)
Weights <- read.csv(file="Weights.csv")
MC <- MetaCommunity(Abundances, Weights)
```

Value

An object of class **MetaCommunity** is a list:

`Nsi`	A matrix containing abundance data, species in line, communities in column.
`Ns`	A vector containing the number of individuals of each species.
`Ni`	A vector containing the number of individuals of each community.
`N`	The total number of individuals.
`Psi`	A matrix whose columns are the probability vectors of communities (each of them sums to 1).
`Wi`	A vector containing the normalized community weights (sum to 1).
`Ps`	A vector containing the probability vector of the metacommunity.
`Nspecies`	The number of species.

`Ncommunities`	The number of communities.
`SampleCoverage`	The sample coverage of the metacommunity.
`SampleCoverage.communities`	A vector containing the sample coverages of each community.

`is.MetaCommunity` returns `TRUE` if the object is of class `MetaCommunity`.

`summary.MetaCommunity` returns a summary of the object's value.

`plot.MetaCommunity` plots it.

Examples

```
# Use BCI data from vegan package
if (require(vegan, quietly = TRUE)) {
  # Load BCI data (number of trees per species in each 1-ha plot of a tropical forest)
  data(BCI)
  # BCI dataframe must be transposed (its lines are plots, not species)
  BCI.df <- as.data.frame(t(BCI))
  # Create a metacommunity object from a matrix of abundances and a vector of weights
  # (here, all plots have a weight equal to 1)
  MC <- MetaCommunity(BCI.df)
}
```

`Optimal.Similarity` *Optimal scale parameter to transform a distance matrix into a similarity matrix*

Description

Calculates the scale parameter u that maximizes the variance of the similarity matrix $exp(-u * DistanceMatrix)$.

Usage

```
Optimal.Similarity(Distance, CheckArguments = TRUE)
```

Arguments

`Distance`	A distance matrix, *i.e.* a square matrix with zeros on its diagonal or a `dist` object.
`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

The similarity matrix used by `Dqz`) can be optimized following Marcon *et al.* (2014) such that the variance of similarities between pairs of species is maximized. See the example.

Value

A list:

`u`	The optimal scale u.
`Matrix`	The optimal similarity matrix `Z`.

References

Marcon, E., Zhang, Z. and Herault, B. (2014). The decomposition of similarity-based diversity and its bias correction. *HAL* hal-00989454(version 3).

See Also

`Dqz`

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest)
data(Paracou618)
# Prepare the similarity matrix. The square root of Paracou618.dist is euclidean.
optimal <- Optimal.Similarity(sqrt(Paracou618.dist))
# Optimal scale
optimal$u
# Calculate diversity of order 2
bcDqz(Paracou618.MC$Ns, 2, optimal$Matrix)
```

`Paracou618.dist` *Functional distances between pairs of species of Paracou field station plots 6 and 18, two 1-ha plots inventoried by the Bridge project.*

Description

This dataset is from Paracou field station, French Guiana, managed by Cirad. Traits are detailed in Marcon and Herault (2014), the distance matrix was built following Paine et al. (2011).

Usage

```
data(Paracou618)
```

Format

An object of class `dist`.

Source

Permanent data census of Paracou.

References

Gourlet-Fleury, S., Guehl, J. M. and Laroussinie, O., Eds. (2004). *Ecology & management of a neotropical rainforest. Lessons drawn from Paracou, a long-term experimental research site in French Guiana*. Paris, Elsevier.

Marcon, E., Herault, B. (2015). Decomposing Phylodiversity. *Methods in Ecology and Evolution* 6(3): 333-339.

Paine, C. E. T., Baraloto, C., Chave, J., and Herault, B. (2011). Functional traits of individual trees reveal ecological constraints on community assembly in tropical rain forests. *Oikos*, 120(5), 720-727.

Examples

```
data(Paracou618)
plot(density(Paracou618.dist, from=0), main="Distances between species")
```

`Paracou618.Functional` *Functional tree of species of Paracou field station plots 6 and 18, two 1-ha plots inventoried by the Bridge project.*

Description

This dataset is from Paracou field station, French Guiana, managed by Cirad. Traits are detailed in Marcon and Herault (2014), the tree was built following Paine et al. (2011), based on `Paracou618.dist`.

Usage

```
data(Paracou618)
```

Format

An object of class `hclust`.

Source

Permanent data census of Paracou.

References

Gourlet-Fleury, S., Guehl, J. M. and Laroussinie, O., Eds. (2004). *Ecology & management of a neotropical rainforest. Lessons drawn from Paracou, a long-term experimental research site in French Guiana*. Paris, Elsevier.

Marcon, E., Herault, B. (2015). Decomposing Phylodiversity. *Methods in Ecology and Evolution* 6(3): 333-339.

Paine, C. E. T., Baraloto, C., Chave, J., and Herault, B. (2011). Functional traits of individual trees reveal ecological constraints on community assembly in tropical rain forests. *Oikos*, 120(5), 720-727.

Examples

```
data(Paracou618)
plot(Paracou618.Functional)
```

`Paracou618.MC` *Paracou field station plots 6 and 18, two 1-ha plots inventoried by the Bridge project.*

Description

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

Usage

```
data(Paracou618)
```

Format

An object of class `MetaCommunity` made of two communities and 425 species.

Source

Permanent data census of Paracou and Marcon et al. (2012).

References

Gourlet-Fleury, S., Guehl, J. M. and Laroussinie, O., Eds. (2004). *Ecology & management of a neotropical rainforest. Lessons drawn from Paracou, a long-term experimental research site in French Guiana*. Paris, Elsevier.

Marcon, E., F. Puech, et al. (2012). Characterizing the relative spatial structure of point patterns. *International Journal of Ecology* 2012(Article ID 619281): 11.

Examples

```
data(Paracou618)
summary(Paracou618.MC)
```

`Paracou618.Taxonomy` *Taxonomy (Family - Genus - Species) of Paracou field station plots 6 and 18, two 1-ha plots inventoried by the Bridge project.*

Description

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

Usage

```
data(Paracou618)
```

Format

An object of class "phylo" (see `read.tree`) containing a taxonomy.

Source

Permanent data census of Paracou.

References

Gourlet-Fleury, S., Guehl, J. M. and Laroussinie, O., Eds. (2004). *Ecology & management of a neotropical rainforest. Lessons drawn from Paracou, a long-term experimental research site in French Guiana*. Paris, Elsevier.

Examples

```
data(Paracou618)
plot(Paracou618.Taxonomy, type="fan", show.tip.label=FALSE)
```

`PDFD` *Phylogenetic Diversity / Functional Diversity of a Community*

Description

Calculates Faith's PD / Petchey and Gaston' FD of a community described by a probability vector and a phylogenetic / functional tree.

Usage

```
PDFD(Ps, Tree, CheckArguments = TRUE)
```

Arguments

`Ps`	A probability vector, summing to 1.
`Tree`	An object of class `hclust`, "phylo" (see `read.tree`), `phylog` or `PPtree`. The tree must be ultrametric.
`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

PD and FD are defined as the total legth of the branches of the tree.

The probability vector is used to select branches: branches with probability 0 are eliminated.

Bias correction requires the number of individuals to estimate sample `Coverage`.

Use `bcPhyloDiversity(Ps, 0, Tree)` and choose the `Correction`.

Value

A named number equal to the calculated diversity. The name is that of the bias correction used.

References

Faith, D. P. (1992). Conservation evaluation and phylogenetic diversity. *Biological Conservation* 61(1): 1-10.

Petchey, O. L. and Gaston, K. J. (2002). Functional diversity (FD), species richness and community composition. *Ecology Letters* 5: 402-411.

See Also

`bcPhyloDiversity`

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest
#      and their taxonomy)
data(Paracou618)
# Ps is the vector of probabilities
Ps <- Paracou618.MC$Ps
# Calculate the phylogenetic Shannon diversity of the plot
PDFD(Ps, Paracou618.Taxonomy)
```

`PhyloApply` *Apply a Function over a Phylogenetic Tree*

Description

Cuts the tree into slices separated by nodes, applies the function to each slice and returns the weighted (by slice lengths) sum of the results.

Usage

```
PhyloApply(Tree, FUN, NorP, Normalize = TRUE, dfArgs = NULL,
           ..., CheckArguments = TRUE)
```

Arguments

`Tree`	An object of class `hclust`, "phylo" (see `read.tree`), `phylog` or `PPtree`. The tree must be ultrametric.
`FUN`	The function to be applied to each interval of the tree.
`NorP`	A numeric vector or a two-column matrix. Contains either abundances or probabilities. Two-column matrices should contain the observed abundances (or probabilities) in the first column and the expected ones in the second column, to allow using beta diversity functions.
`Normalize`	If `TRUE` (default), the `Total` value returned by `Function` is normalized by the height of the tree (it is the weighted average value of the result in each slice). If `FALSE`, it is the unnormalized weighted sum of the results.
`dfArgs`	A dataframe. Columns are arguments for `FUN`: their names are those of valid arguments. Values will be passed to `FUN` in each slice of the tree, starting from the tips. The number of lines must equal the number of slices.
`...`	Further arguments to pass to `Function`.
`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

This funtion is generally not used directly. It is a tool to calculate `PhyloEntropy` and `PhyloDiversity`.

Intervals (slices) separate two cuts (nodes) in a tree: no node is found at heights contained in an interval.

Objects of class **PPtree** are returned by `Preprocess.Tree`.

... allow passing arguments to the function but they can't change along the tree. If necessary, `dfArgs` allow passing a different value for each slice of the tree.

Value

An object of class `PhyloValue`. It is a list:

`Distribution`	The distribution used to calculate the value
`Function`	The function used to calculate the value
`Tree`	The functional or phylogenetic tree used to calculate the value
`Normalized`	Logical. Indicates whether phylovalue is normalized or proportional to the height of the tree.
`Cuts`	A named vector containing values along the tree. Names are cut ends, *i.e.* the ends of intervals (the first interval starts at 0 for leaves, the max value is the height of the tree).
`Corrections`	A named vector containing the correction used by FUN to obtain each value of `Cuts`. Names are those of `Cuts`.
`Total`	The total value, multiplied by the tree height if `Normalize` is FALSE.

References

Marcon, E., Herault, B. (2015). Decomposing Phylodiversity. *Methods in Ecology and Evolution* 6(3): 333-339.

See Also

`Preprocess.Tree`

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest
#      and their taxonomy)
data(Paracou618)
# Plot the taxonomy
plot(Paracou618.Taxonomy, type="fan", show.tip.label=FALSE)
# Calculate the mean number of trees (individuals) per species
# (Cuts are 1=species, 2=genus, 3=family)
summary(PhyloApply(Paracou618.Taxonomy, mean, Paracou618.MC$Ns, TRUE))
```

`PhyloBetaEntropy` *Phylogenetic Beta Entropy of a community*

Description

Calculates the phylogenetic beta entropy of order q of a a community belonging to a metacommunity.

Usage

```
PhyloBetaEntropy(NorP, NorPexp = NULL, q = 1, Tree, Normalize = TRUE, ...)
bcPhyloBetaEntropy(Ns, Nexp, q = 1, Tree, Normalize = TRUE,
  Correction = "Best", CheckArguments = TRUE)
## S3 method for class 'ProbaVector'
PhyloBetaEntropy(NorP, NorPexp = NULL, q = 1, Tree, Normalize = TRUE,
  ..., CheckArguments = TRUE, Ps = NULL, Pexp = NULL)
## S3 method for class 'AbdVector'
PhyloBetaEntropy(NorP, NorPexp = NULL, q = 1, Tree, Normalize = TRUE,
  Correction = "Best", ..., CheckArguments = TRUE, Ns = NULL, Nexp = NULL)
## S3 method for class 'integer'
PhyloBetaEntropy(NorP, NorPexp = NULL, q = 1, Tree, Normalize = TRUE,
  Correction = "Best", ..., CheckArguments = TRUE, Ns = NULL, Nexp = NULL)
## S3 method for class 'numeric'
PhyloBetaEntropy(NorP, NorPexp = NULL, q = 1, Tree, Normalize = TRUE,
  Correction = "Best", ..., CheckArguments = TRUE, Ps = NULL, Ns = NULL,
  Pexp = NULL, Nexp = NULL)
```

Arguments

`Ps`	The probability vector of species of the community.
`Pexp`	The probability vector of species of the metacommunity.
`Ns`	A numeric vector containing species abundances of the community.
`Nexp`	A numeric vector containing species abundances of the metacommunity.
`NorP`	A numeric vector, an integer vector, an abundance vector (`AbdVector`) or a probability vector (`ProbaVector`). Contains either abundances or probabilities of the community.
`NorPexp`	A numeric vector, an integer vector, an abundance vector (`AbdVector`) or a probability vector (`ProbaVector`). Contains either abundances or probabilities of the metacommunity.
`q`	A number: the order of entropy. Default is 1.
`Tree`	An object of class `hclust`, "phylo" (see `read.tree`), `phylog` or `PPtree`. The tree must be ultrametric.
`Normalize`	If `TRUE` (default), the entropy returned by the function is normalized by the height of the tree (it is the weighted average value of the entropy in each slice). If `FALSE`, it is the unnormalized weighted sum of the results.
`Correction`	A string containing one of the possible corrections: currently, only `"ChaoShen"`. `"Best"` is the default value, it is equivalent to `"ChaoShen"`.
`...`	Additional arguments. Unused.
`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

The phylogenetic entropy is the generalization of HCDT entropy to unequal species distances (Pavoine et al., 2009).

Calculation relies on `TsallisBeta` and `PhyloApply`.

Bias correction requires the number of individuals to estimate sample `Coverage`. Use `bcPhyloBetaEntropy` and choose the `Correction`.

Note that beta entropy value is related to alpha entropy (if q is not 1) and cannot be compared accross communities (Jost, 2007). Beta entropy of a community is not meaningful in general, do rather calculate the `PhyloDiversity` of the metacommunity.

The functions are designed to be used as simply as possible. `PhyloBetaEntropy` is a generic method. If its first argument is an abundance vector, an integer vector or a numeric vector which does not sum to 1, the bias corrected function `bcPhyloBetaEntropy` is called. Explicit calls to `bcPhyloBetaEntropy` (with bias correction) or to `PhyloBetaEntropy.ProbaVector` (without correction) are possible to avoid ambiguity. The `.integer` and `.numeric` methods accept `Ps` or `Ns` arguments instead of `NorP` for backward compatibility.

Value

A `PhyloEntropy` object containing entropy values at each cut of the tree.

References

Jost (2007), Partitioning diversity into independent alpha and beta components. *Ecology* 88(10): 2427-2439.

Marcon, E., Herault, B. (2015). Decomposing Phylodiversity. *Methods in Ecology and Evolution* 6(3): 333-339.

Pavoine, S., Love, M. S. and Bonsall, M. B. (2009). Hierarchical partitioning of evolutionary and ecological patterns in the organization of phylogenetically-structured species assemblages: Application to rockfish (genus: Sebastes) in the Southern California Bight. *Ecology Letters* 12(9): 898-908.

See Also

`TsallisBeta`, `bcPhyloBetaEntropy`, `PhyloDiversity`

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest
#      and their taxonomy)
data(Paracou618)
# Ps is the vector of probabilities
Ps <- as.ProbaVector(Paracou618.MC$Ps)
# Probability distribution of the first plot
Ps1 <- as.ProbaVector(Paracou618.MC$Psi[, 1])
# Calculate the phylogenetic Shannon beta entropy of the plot
summary(PhyloBetaEntropy(Ps1, Ps, 1, Paracou618.Taxonomy) -> e)
plot(e)

# Ns is the vector of abundances of the metacommunity
Ns <- as.AbdVector(Paracou618.MC$Ns)
# Abundances in the first plot
Ns1 <- as.AbdVector(Paracou618.MC$Nsi[, 1])
```

```
# Calculate the phylogenetic Shannon beta entropy of the plot
summary(bcPhyloBetaEntropy(Ns1, Ns, 1, Paracou618.Taxonomy, Correction = "Best") -> e)
plot(e)
```

`PhyloDiversity` *Phylogenetic Diversity of a Community*

Description

Calculates the phylogenetic diversity of order q of a probability vector.

Usage

```
PhyloDiversity(NorP, q = 1, Tree, Normalize = TRUE, ...)
bcPhyloDiversity(Ns, q = 1, Tree, Normalize = TRUE, Correction = "Best",
  CheckArguments = TRUE)
## S3 method for class 'ProbaVector'
PhyloDiversity(NorP, q = 1, Tree, Normalize = TRUE,
  ..., CheckArguments = TRUE, Ps = NULL)
## S3 method for class 'AbdVector'
PhyloDiversity(NorP, q = 1, Tree, Normalize = TRUE,
  Correction = "Best", ..., CheckArguments = TRUE, Ns = NULL)
## S3 method for class 'integer'
PhyloDiversity(NorP, q = 1, Tree, Normalize = TRUE,
  Correction = "Best", ..., CheckArguments = TRUE, Ns = NULL)
## S3 method for class 'numeric'
PhyloDiversity(NorP, q = 1, Tree, Normalize = TRUE,
  Correction = "Best", ..., CheckArguments = TRUE, Ps = NULL, Ns = NULL)
is.PhyloDiversity(x)
## S3 method for class 'PhyloDiversity'
summary(object, ...)
```

Arguments

`Ps`	A probability vector, summing to 1.
`Ns`	A numeric vector containing species abundances.
`NorP`	A numeric vector, an integer vector, an abundance vector (`AbdVector`) or a probability vector (`ProbaVector`). Contains either abundances or probabilities.
`q`	A number: the order of diversity. Default is 1.
`Tree`	An object of class `hclust`, "phylo" (see `read.tree`), `phylog` or `PPtree`. The tree must be ultrametric.
`Normalize`	If `TRUE` (default), the `Total` diversity is not affected by the height of the tree. If `FALSE`, it is proportional to the height of the tree.

`Correction`	A string containing one of the possible corrections: `"None"` (no correction), `"ChaoShen"`, `"Grassberger"`, `"Holste"`, `"Bonachela"` or `"Best"`, the default value.
`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.
`x`	An object to be tested or plotted
`object`	A `PhyloDiversity` object to be summarized.
`...`	Additional arguments to be passed to the generic methods.

Details

The phylogenetic entropy is its generalization of HCDT entropy to unequal species distances (Pavoine et al., 2009).

Diversity is obtained by transforming generalized entropy.

Bias correction requires the number of individuals to estimate sample `Coverage`. Use `bcPhyloDiversity` and choose the `Correction`.

The functions are designed to be used as simply as possible. `PhyloDiversity` is a generic method. If its first argument is an abundance vector, an integer vector or a numeric vector which does not sum to 1, the bias corrected function `bcPhyloDiversity` is called. Explicit calls to `bcPhyloDiversity` (with bias correction) or to `PhyloDiversity.ProbaVector` (without correction) are possible to avoid ambiguity. The `.integer` and `.numeric` methods accept `Ps` or `Ns` arguments instead of `NorP` for backward compatibility.

Value

An object of class `PhyloDiversity` is a list:

`Distribution`	The distribution used to calculate diversity
`Function`	The function used to calculate diversity
`Tree`	The functional or phylogenetic tree used to calculate diversity
`Normalized`	Logical. Indicates whether phylodiversity is normalized or proportional to the height of the tree.
`Type`	The type of diversity ("alpha", "beta" or "gamma").
`Order`	The order of diversity q.
`Cuts`	A named vector containing values of neutral diversity along the tree. Names are cut ends, *i.e.* the ends of intervals (the first interval starts at 0 for leaves, the max value is the height of the tree).
`Total`	A value equal the total diversity (obtained by transforming the total normalized entropy), multiplied by the tree height if `Normalize` is FALSE.

`is.PhyloDiversity` returns `TRUE` if the object is of class `PhyloDiversity`.

`summary.PhyloDiversity` returns a summary of the object's value.

`PhyloDiversity` objects can be plotted by `plot.PhyloValue` because `PhyloDiversity` objects are also of class `PhyloValue`.

Note

The tree must contain all species of the probability vector. If it contains extra species, computation time will just be increased.

References

Chao, A., Chiu, C.-H. and Jost, L. (2010). Phylogenetic diversity measures based on Hill numbers. *Philosophical Transactions of the Royal Society B* 365(1558): 3599-609.

Marcon, E., Herault, B. (2015). Decomposing Phylodiversity. *Methods in Ecology and Evolution* 6(3): 333-339.

Pavoine, S., Love, M. S. and Bonsall, M. B. (2009). Hierarchical partitioning of evolutionary and ecological patterns in the organization of phylogenetically-structured species assemblages: Application to rockfish (genus: Sebastes) in the Southern California Bight. *Ecology Letters* 12(9): 898-908.

See Also

`PhyloEntropy`, `Diversity`

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest
#      and their taxonomy)
data(Paracou618)
# Ps is the vector of probabilities
Ps <- as.ProbaVector(Paracou618.MC$Ps)
# Calculate the phylogenetic Shannon diversity of the plot
summary(PhyloDiversity(Ps, 1, Paracou618.Taxonomy) -> d)
plot(d)

# Ns is the vector of abundances of the metacommunity
Ns <- as.AbdVector(Paracou618.MC$Ns)
# Calculate the phylogenetic Shannon diversity of the plot
summary(bcPhyloDiversity(Ns, 1, Paracou618.Taxonomy, Correction = "Best") -> d)
plot(d)
```

`PhyloEntropy` *Phylogenetic Entropy of a community*

Description

Calculates the phylogenetic entropy of order q of a probability vector.

Usage

```
PhyloEntropy(NorP, q = 1, Tree, Normalize = TRUE, ...)
bcPhyloEntropy(Ns, q = 1, Tree, Normalize = TRUE, Correction = "Best",
  SampleCoverage = NULL, CheckArguments = TRUE)
## S3 method for class 'ProbaVector'
PhyloEntropy(NorP, q = 1, Tree, Normalize = TRUE,
  ..., CheckArguments = TRUE, Ps = NULL)
## S3 method for class 'AbdVector'
PhyloEntropy(NorP, q = 1, Tree, Normalize = TRUE, Correction = "Best",
  ..., CheckArguments = TRUE, Ns = NULL)
## S3 method for class 'integer'
PhyloEntropy(NorP, q = 1, Tree, Normalize = TRUE, Correction = "Best",
  ..., CheckArguments = TRUE, Ns = NULL)
## S3 method for class 'numeric'
PhyloEntropy(NorP, q = 1, Tree, Normalize = TRUE, Correction = "Best",
  ..., CheckArguments = TRUE, Ps = NULL, Ns = NULL)
is.PhyloEntropy(x)
## S3 method for class 'PhyloEntropy'
summary(object, ...)
```

Arguments

`Ps`	A probability vector, summing to 1.
`Ns`	A numeric vector containing species abundances.
`NorP`	A numeric vector, an integer vector, an abundance vector (`AbdVector`) or a probability vector (`ProbaVector`). Contains either abundances or probabilities.
`q`	A number: the order of entropy. Default is 1.
`Tree`	An object of class `hclust`, "phylo" (see `read.tree`), `phylog` or `PPtree`. The tree must be ultrametric.
`Normalize`	If `TRUE` (default), the `Total` entropy returned by the function is normalized by the height of the tree (it is the weighted average value of the entropy in each slice). If `FALSE`, it is the unnormalized weighted sum of the results.
`Correction`	A string containing one of the possible corrections supported by `Tsallis`.
`SampleCoverage`	The sample coverage of `Ns` calculated elsewhere. Used to calculate the gamma diversity of meta-communities, see details.
`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.
`x`	An object to be tested or plotted
`object`	A `PhyloEntropy` object to be summarized.
`...`	Additional arguments to be passed to the generic methods.

Details

The phylogenetic entropy is its generalization of HCDT entropy to unequal species distances (Pavoine et al., 2009).

Calculation relies on `Tsallis` and `PhyloApply`.

Intervals separate two cuts in a tree: no node is found at heights contained in an interval.

Bias correction requires the number of individuals to estimate sample `Coverage`. Use `bcPhyloEntropy` and choose the `Correction`.

The functions are designed to be used as simply as possible. `PhyloEntropy` is a generic method. If its first argument is an abundance vector, an integer vector or a numeric vector which does not sum to 1, the bias corrected function `bcPhyloEntropy` is called. Explicit calls to `bcPhyloEntropy` (with bias correction) or to `PhyloEntropy.ProbaVector` (without correction) are possible to avoid ambiguity. The `.integer` and `.numeric` methods accept `Ps` or `Ns` arguments instead of `NorP` for backward compatibility.

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 `SampleCoverage` argument allows applying the `"ChaoShen"` and `"Grassberger"` corrections to estimate quite well the entropy. `DivPart` and `GammaEntropy` functions use this tweak.

Value

An object of class `PhyloEntropy` is a list:

`Distribution`	The distribution used to calculate entropy
`Function`	The function used to calculate entropy
`Tree`	The functional or phylogenetic tree used to calculate entropy
`Normalized`	Logical. Indicates whether phyloentropy is normalized or proportional to the height of the tree.
`Type`	The type of entropy ("alpha", "beta" or "gamma").
`Order`	The order of entropy q.
`Cuts`	A named vector containing values of neutral entropy along the tree. Names are cut ends, *i.e.* the ends of intervals (the first interval starts at 0 for leaves, the max value is the height of the tree).
`Total`	A value equal the total entropy multiplied by the tree height if `Normalize` is FALSE.

`is.PhyloEntropy` returns `TRUE` if the object is of class `PhyloEntropy`.

`summary.PhyloEntropy` returns a summary of the object's value.

`PhyloEntropy` objects can be plotted by `plot.PhyloValue` because `PhyloEntropy` objects are also of class `PhyloValue`.

Note

The tree must contain all species of the probability vector. If it contains extra species, computation time will just be increased.

References

Marcon, E., Herault, B. (2015). Decomposing Phylodiversity. *Methods in Ecology and Evolution* 6(3): 333-339.

Pavoine, S., Love, M. S. and Bonsall, M. B. (2009). Hierarchical partitioning of evolutionary and ecological patterns in the organization of phylogenetically-structured species assemblages: Application to rockfish (genus: Sebastes) in the Southern California Bight. *Ecology Letters* 12(9): 898-908.

See Also

`Tsallis`, `PhyloDiversity`

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest
#      and their taxonomy)
data(Paracou618)
# Ps is the vector of probabilities
Ps <- as.ProbaVector(Paracou618.MC$Ps)
# Calculate the phylogenetic Shannon entropy of the plot
summary(PhyloEntropy(Ps, 1, Paracou618.Taxonomy) -> e)
plot(e)

# Ns is the vector of abundances of the metacommunity
Ns <- as.AbdVector(Paracou618.MC$Ns)
# Calculate the phylogenetic Shannon entropy of the plot
summary(bcPhyloEntropy(Ns, 1, Paracou618.Taxonomy, Correction = "Best") -> e)
plot(e)
```

`PhyloValue` *Phylogenetic entropy or diversity.*

Description

Entropy or diversity against the height of the phylogenetic or functional tree.

Usage

```
is.PhyloValue(x)
## S3 method for class 'PhyloValue'
autoplot(object, xlab = expression(italic("T")), ylab = NULL, main = NULL,
          col = "black",
          lty = 1,
          lwd = 0.5,
          ...)
## S3 method for class 'PhyloValue'
plot(x, xlab = expression(italic("T")), ylab = NULL, main = NULL, ...)
## S3 method for class 'PhyloValue'
summary(object, ...)
```

Arguments

`x`	An object of class `PhyloValue`, including `PhyloDiversity` and `PhyloEntropy` objects.
`xlab`	The X axis label, "T" by default for Time.
`ylab`	The Y axis label. if `NULL` (by default), "Entropy" or "Diversity" or nothing is chosen according to the object class.
`main`	The main title of the plot. if `NULL` (by default), a default value is used.
`object`	A `PhyloValue` object to be summarized.
`col`	The color of the geom objects. See "Color Specification" in `par`.
`lty`	The type of the lines. See `lines`.
`lwd`	The width of the lines. See `lines`.
`...`	Additional arguments to be passed to `plot`.

Details

`PhyloValue` objects are the result of `PhyloApply`.

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest
#      and their taxonomy)
data(Paracou618)
# Calculate richness along the tree
# (Cuts are 1=species, 2=genus, 3=family)
summary(r <- PhyloApply(Paracou618.Taxonomy, FUN=Richness,
  NorP=Paracou618.MC$Ns, Normalize=TRUE))
autoplot(r)
```

`PPtree` *Preprocessed Trees.*

Description

Methods for objects of type "`PPtree`".

Usage

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

Arguments

`x`	An object to be tested or plotted
`...`	Additional arguments to be passed to the generic methods.

Value

An object of class **PPtree** is a list:

`phyTree`	A "phylo" (see `read.tree`) tree
`hTree`	A `hclust` tree
`Height`	The height of the tree, that is to say the distance between root and leaves
`Cuts`	A vector. Cut times of the tree (the distance from nodes to leaves)
`Intervals`	A vector. The lengths of intervals between cuts

`is.PPtree` returns `TRUE` if the object is of class `PPtree`.

`plot.PPtree` plots it.

Note

Versions up to 1.3 contained a `phylog` tree, now deprecated in `ade4`. A "phylo" (see `read.tree`) tree is now used.

See the dedicated vignette (`vignette("Phylogenies", package="entropart")`) for more details.

Examples

```
data(Paracou618)
# Preprocess a phylog object
ppt <- Preprocess.Tree(EightSpTree)
# Is it a preprocessed tree?
is.PPtree(ppt)
# Plot it
plot(ppt, hang=-1)
# Alternative plot
ade4::radial.phylog(EightSpTree)
```

`RAC` *Fit Distributions to Well-Known Rank Abundance Curves.*

Description

Observed distributions are fitted to classical RAC's.

Usage

```
RAClnorm(Ns, CheckArguments = TRUE)
RACgeom(Ns, CheckArguments = TRUE)
RAClseries(Ns, CheckArguments = TRUE)
RACbstick(Ns, CheckArguments = TRUE)
```

Arguments

`Ns`	A numeric vector containing species abundances.
`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

`SpeciesDistribution` or integer vectors can be used to fit classical rank-abundance curves (RAC) of classical distributions: `"RAClnorm"` for log-normal (Preston, 1948), `"RAClseries"` for log-series (Fisher *et al.*, 1943), `"RACgeom"` for geometric (Motomura, 1932) or `"RACbstick"` for broken stick (MacArthur, 1957). method returns the estimated parameters of the fitted distribution. The broken stick has no parameter, so the maximum abundance is returned.

Value

A list (the parameters of distributions are returned only if the distribution has been fit):

`Rank`	A numeric vector. The ranks of species in the fitted RAC.
`Abundance`	The abundance of species in the fitted RAC.
`mu`	The expectation of the log-normal distribution
`sigma`	The standard deviation of the log-normal distribution
`alpha`	Fisher's alpha in the log-series distribution
`prob`	The proportion of ressources taken by successive species in the geometric distribution
`max`	The maximum abundance in the broken-stick distribution

Note

Fisher's alpha is estimated to fit the log-series distribution. The estimation is done by the `fisher.alpha` function of package `vegan`. It may differ substantially from the estimation returned by `optimal.theta` from package `untb`.

Author(s)

Eric Marcon <Eric.Marcon@agroparistech.fr>, Bruno Herault <Bruno.Herault@cirad.fr>

References

Fisher R.A., Corbet A.S., Williams C.B. (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.

MacArthur R.H. (1957) On the Relative Abundance of Bird Species. *PNAS* 43(3): 293-295.

Motomura I. (1932) On the statistical treatment of communities. *Zoological Magazine* 44: 379-383.

Preston, F.W. (1948). The commonness, and rarity, of species. *Ecology* 29(3): 254-283.

See Also

`rgeom`, `rlnorm`, `rCommunity`, `plot.SpeciesDistribution`

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest)
data(Paracou618)
# Ns is the total number of trees per species
Ns <- as.AbdVector(Paracou618.MC$Ns)
# Fitted parameters
RACln <- RAClnorm(Ns)
RACln$mu
RACln$sigma
RACgeom(Ns)$prob
RAClseries(Ns)$alpha
RACbstick(Ns)$max
```

`Rao` *Rao Quadratic Entropy of a Community*

Description

Calculates Rao's quadratic entropy of a community described by a probability vector and a phylogenetic / functional tree.

Usage

```
Rao(NorP, Tree, ...)
bcRao(Ns, Tree, Correction="Lande", CheckArguments = TRUE)
## S3 method for class 'ProbaVector'
Rao(NorP, Tree, ..., CheckArguments = TRUE, Ps = NULL)
## S3 method for class 'AbdVector'
Rao(NorP, Tree, Correction = "Lande", ...,
  CheckArguments = TRUE, Ns = NULL)
## S3 method for class 'integer'
Rao(NorP, Tree, Correction = "Lande", ...,
  CheckArguments = TRUE, Ns = NULL)
## S3 method for class 'numeric'
Rao(NorP, Tree, Correction = "Lande", ...,
  CheckArguments = TRUE, Ps = NULL, Ns = NULL)
```

Arguments

`Ps`	A probability vector, summing to 1.
`Ns`	A numeric vector containing species abundances.
`NorP`	A numeric vector, an integer vector, an abundance vector (`AbdVector`) or a probability vector (`ProbaVector`). Contains either abundances or probabilities.

`Tree`	An object of class `hclust`, "phylo" (see `read.tree`), `phylog` or `PPtree`. The tree must be ultrametric.
`Correction`	A string containing one of the possible corrections accepted by `bcTsallis` or `"Lande"`, the default value (equivalent to `"Best"`).
`...`	Additional arguments. Unused.
`CheckArguments`	Logical; 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. Use `bcRao` and choose the `Correction`.

The unbiased estimator of Rao's entropy is identical to that of Simpson's entropy because Rao's entropy is a linear sum of Simson entropies, all of them calculated from the same number of individuals (Marcon and Herault, 2014). It equals the plug-in etimator multiplied by n/(n-1) where n is the total number of individuals.

The functions are designed to be used as simply as possible. `Tsallis` is a generic method. If its first argument is an abundance vector, an integer vector or a numeric vector which does not sum to 1, the bias corrected function `bcTsallis` is called. Explicit calls to `bcTsallis` (with bias correction) or to `Tsallis.ProbaVector` (without correction) are possible to avoid ambiguity. The `.integer` and `.numeric` methods accept `Ps` or `Ns` arguments instead of `NorP` for backward compatibility.

Value

A named number equal to the calculated entropy. The name is that of the bias correction used.

References

Marcon, E., Herault, B. (2015). Decomposing Phylodiversity. *Methods in Ecology and Evolution* 6(3): 333-339.

Rao, C. R. (1982). Diversity and dissimilarity coefficients: a unified approach. *Theoretical Population Biology* 21: 24-43.

See Also

`bcPhyloDiversity`

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest)
data(Paracou618)
# Ns is the total number of trees per species
Ns <- as.AbdVector(Paracou618.MC$Ns)
# Species probabilities
Ps <- as.ProbaVector(Paracou618.MC$Ns)
# Calculate Rao's quadratic entropy of the plot
Rao(Ps, Paracou618.Taxonomy)
```

`rCommunity` *Random Communities*

Description

Draws random communities according to a probability distribution.

Usage

```
rCommunity(n, size = sum(NorP), NorP = 1, BootstrapMethod = "Chao2015", S = 300,
    Distribution = "lnorm",  sd = 1, prob = 0.1, alpha = 40,
    CheckArguments = TRUE)
```

Arguments

`n`	The number of communities to draw.
`size`	The number of individuals to draw in each community.
`BootstrapMethod`	The method used to obtain the probabilities to generate bootstrapped communities from observed abundances. If `"Marcon"`, 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.ProbaVector`) following Chao *et al.* (2013) or Chao *et al.* (2015).
`NorP`	A numeric vector. Contains either abundances or probabilities.
`S`	The number of species.
`Distribution`	The distribution of species frequencies. May be `"lnorm"` (log-normal), `"lseries"` (log-series), `"geom"` (geometric) or `"bstick"` (broken stick).
`sd`	The simulated distribution standard deviation. For the log-normal distribution, this is the standard deviation on the log scale.
`prob`	The proportion of ressources taken by successive species.
`alpha`	Fisher's alpha.
`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

Communities of fixed `size` are drawn in a multinomial distribution according to the distribution of probabilities provided by `NorP`.

An abundance vector may be used instead of probabilities, then `size` is by default the total number of individuals in the vector. Random communities are built by drawing in a multinomial law following Marcon *et al.* (2012), or trying to estimate the distribution of the actual community with `as.ProbaVector`. If `BootstrapMethod = "Chao2013"`, the distribution is estimated by a single parameter model and unobserved species are given equal probabilities. If `BootstrapMethod =`

`"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 a lognormal (`"lnorm"`) one (Preston, 1948) with given standard deviation (`sd`; note that the mean is actually a normalizing constant. Its values is set equal to 0 for the simulation of the normal distribution of unnormalized log-abundances), a log-series (`"lseries"`) one (Fisher *et al.*, 1943) with parameter `alpha`, a geometric (`"geom"`) one (Motomura, 1932) with parameter `prob`, or a broken stick (`"bstick"`) one (MacArthur, 1957). The number of simulated species is fixed by `S`, except for `"lseries"` where it is obtained from `alpha` and `size`: $S = \alpha \ln(1 + \frac{size}{\alpha})$.

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

Value

A vector of species abundances (`AbdVector`) if a single community has been drawn, or a `MetaCommunity` containing simulated communities.

References

Chao, A., Wang, Y. T. and 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.

Chao, A., Hsieh, T. C., Chazdon, R. L., Colwell, R. K., Gotelli, N. J. (2015) Unveiling the Species-Rank Abundance Distribution by Generalizing Good-Turing Sample Coverage Theory. *Ecology* 96(5): 1189-1201.

Fisher R.A., Corbet A.S., Williams C.B. (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.

MacArthur R.H. (1957) On the Relative Abundance of Bird Species. *PNAS* 43(3): 293-295.

Marcon, E., Herault, B., Baraloto, C. and Lang, G. (2012). The Decomposition of Shannon's Entropy and a Confidence Interval for Beta Diversity. *Oikos* 121(4): 516-522.

Motomura I. (1932) On the statistical treatment of communities. *Zoological Magazine* 44: 379-383.

Preston, F.W. (1948). The commonness, and rarity, of species. *Ecology* 29(3): 254-283.

Reese G. C., Wilson K. R., Flather C. H. (2013) Program SimAssem: Software for simulating species assemblages and estimating species richness. *Methods in Ecology and Evolution* 4: 891-896.

See Also

`SpeciesDistribution` and the program `SimAssem` (Reese *et al.*, 2013; not an R package) for more distributions.

Examples

```
# Generate communities made of 100000 individuals among 300 species and fit them
par(mfrow = c(2,2))
for (d in c("lnorm", "lseries", "geom", "bstick")) {
```

```
rCommunity(n = 1, size = 1E5, S = 300, Distribution = d) -> AbdVec
plot(AbdVec, Distribution = d, main = d)
}
```

`Richness` *Number of species of a community*

Description

Calculates the number of species from probability vector. The name is that of the estimator (the bias correction) used.

Usage

```
Richness(NorP, ...)
bcRichness(Ns, Correction = "Best", Alpha = 0.05, JackOver = FALSE, JackMax = 10,
              CheckArguments = TRUE)
## S3 method for class 'ProbaVector'
Richness(NorP, ..., CheckArguments = TRUE, Ps = NULL)
## S3 method for class 'AbdVector'
Richness(NorP, Correction = "Best", Alpha = 0.05,
  JackOver = FALSE, JackMax = 10,
 Level = NULL, PCorrection = "Chao2015", Unveiling = "geom", RCorrection = "Rarefy",
  ..., CheckArguments = TRUE, Ns = NULL)
## S3 method for class 'integer'
Richness(NorP, Correction = "Best", Alpha = 0.05,
  JackOver = FALSE, JackMax = 10,
 Level = NULL, PCorrection = "Chao2015", Unveiling = "geom", RCorrection = "Rarefy",
  ..., CheckArguments = TRUE, Ns = NULL)
## S3 method for class 'numeric'
Richness(NorP, Correction = "Best", Alpha = 0.05,
  JackOver = FALSE, JackMax = 10,
 Level = NULL, PCorrection = "Chao2015", Unveiling = "geom", RCorrection = "Rarefy",
  ..., CheckArguments = TRUE, Ps = NULL, Ns = NULL)
```

Arguments

`Ps`	A probability vector, summing to 1.
`Ns`	A numeric vector containing species abundances.
`NorP`	A numeric vector, an integer vector, an abundance vector (`AbdVector`) or a probability vector (`ProbaVector`). Contains either abundances or probabilities.
`Correction`	A string containing one of the possible corrections: `"None"` (no correction), `"Jackknife"`, `"iChao1"`, or `"Chao1"`. `"Best"`, the default value, is currently `"Jackknife"`. Ignored by richness interpolation, and by extrapolation if `PCorrection` is not "None".
`Alpha`	The risk level, 5% by default, used to optimize the jackknife order.

`JackOver`	If `TRUE`, retain the jackknife order immediately superior to the optimal one, usually resulting in the overestimation of the number of species. Default is `FALSE`.
`JackMax`	The highest jackknife order allowed. Default is 10. Allowed values are between 1 and 10.
`Level`	The level of interpolation or extrapolation. It may be an a chosen sample size (an integer) or a sample coverage (a number between 0 and 1). Richness extrapolation require its asymptotic estimation depending on the choice of `Correction`.
`PCorrection`	A string containing one of the possible corrections to estimate a probability distribution in `as.ProbaVector`: `"Chao2015"` is the default value. If "None", the asymptotic distribution is not estimated and extrapolation relies only on the asymptotic estimator of richness. Used only for extrapolation.
`Unveiling`	A string containing one of the possible unveiling methods to estimate the probabilities of the unobserved species in `as.ProbaVector`: `"geom"` (the unobserved species distribution is geometric) is the default value. Used only for extrapolation.
`RCorrection`	A string containing a correction recognized by `Richness` to evaluate the total number of species in `as.ProbaVector`. `"Rarefy"` is the default value to estimate the number of species such that the entropy of the asymptotic distribution rarefied to the observed sample size equals the observed entropy of the data. Used only for extrapolation.
`...`	Additional arguments. Unused.
`CheckArguments`	Logical; 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. Use `bcRichness` and choose the `Correction`.

Chao correction techniques are from Chao (1984) and Chiu *et al.* (2015). The Jackknife estimator is calculated by a straight adaptation of the code by Ji-Ping Wang (`jackknife` in CRAN-archived package `SPECIES`). The optimal order is selected according to Burnham and Overton (1978; 1979). The argument `JackOver` allows selecting one order over the optimal one. Many other estimators are available elsewhere, the ones implemented here are necessary for other entropy estimations.

The functions are designed to be used as simply as possible. `Richness` is a generic method. If its first argument is an abundance vector, an integer vector or a numeric vector which does not sum to 1, the bias corrected function `bcRichness` is called.

Richness 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. Extrapolation relies on the estimation of the asymptotic richness. If `PCorrection` is "None", then the asymptotic estimation of richness is made using the chosen `Correction`, else the asymtpotic distribution of the community is derived and its estimated richness adjusted so that the entropy of a sample of this distribution of the size of the actual sample has the entropy of the actual sample.

Value

A named number equal to the estimated number of species. The name is the `Correction`, except for "SAC" (Species Accumulation Curve) for interpolation.

References

Burnham, K. P., and Overton,W. S. (1978), Estimation of the Size of a Closed Population When Capture Probabilities Vary Among Animals. *Biometrika*, 65: 625-633.

Burnham, K. P., and Overton,W. S. (1979), Robust Estimation of Population Size When Capture Probabilities Vary Among Animals. *Ecology* 60:927-936.

Chao, A. (1984) Nonparametric estimation of the number of classes in a population. *Scandinavian Journal of Statistics* 11: 265-270.

Chao, A., Gotelli, N. J., Hsieh, T. C., Sander, E. L., Ma, K. H., Colwell, R. K., Ellison, A. M (2014). Rarefaction and extrapolation with Hill numbers: A framework for sampling and estimation in species diversity studies. *Ecological Monographs*, 84(1): 45-67.

Chiu, C.-H., Wang, Y.-T., Walther, B. A., Chao, A. (2014) An Improved Nonparametric Lower Bound of Species Richness via a Modified Good-Turing Frequency Formula. *Biometrics* 70(3): 671-682.

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest)
data(Paracou618)
# Ns is the total number of trees per species
Ns <- as.AbdVector(Paracou618.MC$Ns)
# Species probabilities
Ps <- as.ProbaVector(Paracou618.MC$Ns)
# Whittaker plot
plot(Ns)

# Number of observed species
Richness(Ps)
# Estimate the actual number of species
bcRichness(Ns, Correction = "Chao1")
bcRichness(Ns, Correction = "iChao1")
bcRichness(Ns, Correction = "Jackknife")
bcRichness(Ns, Correction = "Jackknife", JackOver=TRUE)
```

`Shannon` *Shannon entropy of a community*

Description

Calculates the Shannon entropy of a probability vector.

Usage

```
Shannon(NorP, ...)
bcShannon(Ns, Correction = "Best", CheckArguments = TRUE)
## S3 method for class 'ProbaVector'
Shannon(NorP, ..., CheckArguments = TRUE, Ps = NULL)
## S3 method for class 'AbdVector'
```

```
Shannon(NorP, Correction = "Best", Level = NULL,
  PCorrection = "Chao2015", Unveiling = "geom", RCorrection = "Rarefy", ...,
  CheckArguments = TRUE, Ns = NULL)
## S3 method for class 'integer'
Shannon(NorP, Correction = "Best", Level = NULL,
  PCorrection = "Chao2015", Unveiling = "geom", RCorrection = "Rarefy", ...,
  CheckArguments = TRUE, Ns = NULL)
## S3 method for class 'numeric'
Shannon(NorP, Correction = "Best", Level = NULL,
  PCorrection = "Chao2015", Unveiling = "geom", RCorrection = "Rarefy", ...,
  CheckArguments = TRUE, Ps = NULL, Ns = NULL)
```

Arguments

`Ps`	A probability vector, summing to 1.
`Ns`	A numeric vector containing species abundances.
`NorP`	A numeric vector, an integer vector, an abundance vector (`AbdVector`) or a probability vector (`ProbaVector`). Contains either abundances or probabilities.
`Correction`	A string containing one of the possible asymptotic estimators: `"None"` (no correction), `"ChaoShen"`, `"GenCov"`, `"Grassberger"`, `"Grassberger2003"`, `"Schurmann"`, `"Holste"`, `"Bonachela"`, `"Miller"`, `"ZhangHz"`, `"ChaoJost"`, `"Marcon"`, `"UnveilC"`, `"UnveiliC"`, `"UnveilJ"` or `"Best"`, the default value. Currently, `"Best"` is `"UnveilJ"`.
`Level`	The level of interpolation or extrapolation. It may be an a chosen sample size (an integer) or a sample coverage (a number between 0 and 1). Entropy extrapolation require its asymptotic estimation depending on the choice of `Correction`. Entropy interpolation relies on the estimation of Abundance Frequence Counts: then, `Correction` is passed to `AbdFreqCount` as its `Estimator` argument.
`PCorrection`	A string containing one of the possible corrections to estimate a probability distribution in `as.ProbaVector`: `"Chao2015"` is the default value. Used only for extrapolation.
`Unveiling`	A string containing one of the possible unveiling methods to estimate the probabilities of the unobserved species in `as.ProbaVector`: `"geom"` (the unobserved species distribution is geometric) is the default value. If `"None"`, the asymptotic distribution is not unveiled and only the asymptotic estimator is used. Used only for extrapolation.
`RCorrection`	A string containing a correction recognized by `Richness` to evaluate the total number of species in `as.ProbaVector`. `"Rarefy"` is the default value to estimate the number of species such that the entropy of the asymptotic distribution rarefied to the observed sample size equals the observed entropy of the data. Used only for extrapolation.
`...`	Additional arguments. Unused.
`CheckArguments`	Logical; 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 to estimate sample `Coverage`.

Correction techniques are from Miller (1955), Chao and Shen (2003), Grassberger (1988), Grassberger (2003), Schurmann (2003), Holste *et al.* (1998), Bonachela *et al.* (2008), Zhang (2012), Chao, Wang and Jost (2013). More estimators can be found in the `entropy` package.

Using `MetaCommunity` mutual information, Chao, Wang and Jost (2013) calculate reduced-bias Shannon beta entropy (see the last example below) with better results than the Chao and Shen estimator, but community weights cannot be arbitrary: they must be proportional to the number of individuals.

The functions are designed to be used as simply as possible. `Shannon` is a generic method. If its first argument is an abundance vector, an integer vector or a numeric vector which does not sum to 1, the bias corrected function `bcShannon` is called.

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. Extrapolation relies on the estimation of the asymptotic entropy. If `Unveiling` is "None", then the asymptotic estimation of entropy is made using the chosen `Correction`, else the asymtpotic distribution of the community is derived and its estimated richness adjusted so that the entropy of a sample of this distribution of the size of the actual sample has the entropy of the actual sample.

Value

A named number equal to the calculated entropy. The name is that of the bias correction used.

References

Bonachela, J. A., Hinrichsen, H. and Munoz, M. A. (2008). Entropy estimates of small data sets. *Journal of Physics A: Mathematical and Theoretical* 41(202001): 1-9.

Chao, A. and Shen, T. J. (2003). Nonparametric estimation of Shannon's index of diversity when there are unseen species in sample. *Environmental and Ecological Statistics* 10(4): 429-443.

Chao, A., Wang, Y. T. and 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.

Chao, A., Gotelli, N. J., Hsieh, T. C., Sander, E. L., Ma, K. H., Colwell, R. K., Ellison, A. M (2014). Rarefaction and extrapolation with Hill numbers: A framework for sampling and estimation in species diversity studies. *Ecological Monographs*, 84(1): 45-67.

Grassberger, P. (1988). Finite sample corrections to entropy and dimension estimates. *Physics Letters A* 128(6-7): 369-373.

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

Holste, D., Grosse, I. and Herzel, H. (1998). Bayes' estimators of generalized entropies. *Journal of Physics A: Mathematical and General* 31(11): 2551-2566.

Miller, G. (1955) Note on the bias of information estimates. In: Quastler, H., editor. *Information Theory in Psychology: Problems and Methods*: 95-100.

Shannon, C. E. (1948). A Mathematical Theory of Communication. *The Bell System Technical Journal* 27: 379-423, 623-656.

Schurmann, T. (2004). Bias analysis in entropy estimation. *Journal of Physics A: Mathematical and Theoretical* 37(27): L295-L301.

Tsallis, C. (1988). Possible generalization of Boltzmann-Gibbs statistics. *Journal of Statistical Physics* 52(1): 479-487.

Zhang, Z. (2012). Entropy Estimation in Turing's Perspective. *Neural Computation* 24(5): 1368-1389.

See Also

`bcShannon`, `Tsallis`

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest)
data(Paracou618)
# Ns is the total number of trees per species
Ns <- as.AbdVector(Paracou618.MC$Ns)
# Species probabilities
Ps <- as.ProbaVector(Paracou618.MC$Ns)
# Whittaker plot
plot(Ns)

# Calculate Shannon entropy
Shannon(Ps)

# Calculate the best estimator of Shannon entropy
Shannon(Ns)

# Use metacommunity data to calculate reduced-bias Shannon beta as mutual information
(bcShannon(Paracou618.MC$Ns) + bcShannon(colSums(Paracou618.MC$Nsi))
- bcShannon(Paracou618.MC$Nsi))
```

`ShannonBeta` *Shannon beta entropy of a community*

Description

Calculates the Shannon beta entropy of a community belonging to a metacommunity.

Usage

```
ShannonBeta(NorP, NorPexp = NULL, ...)
bcShannonBeta(Ns, Nexp, Correction = "Best", CheckArguments = TRUE)
## S3 method for class 'ProbaVector'
ShannonBeta(NorP, NorPexp = NULL, ...,
  CheckArguments = TRUE, Ps = NULL, Pexp = NULL)
## S3 method for class 'AbdVector'
ShannonBeta(NorP, NorPexp = NULL, Correction = "Best", ...,
```

```
  CheckArguments = TRUE, Ns = NULL, Nexp = NULL)
## S3 method for class 'integer'
ShannonBeta(NorP, NorPexp = NULL, Correction = "Best", ...,
  CheckArguments = TRUE, Ns = NULL, Nexp = NULL)
## S3 method for class 'numeric'
ShannonBeta(NorP, NorPexp = NULL, Correction = "Best", ...,
  CheckArguments = TRUE, Ps = NULL, Ns = NULL, Pexp = NULL, Nexp = NULL)
```

Arguments

`Ps`	The probability vector of species of the community.
`Pexp`	The probability vector of species of the metacommunity.
`Ns`	A numeric vector containing species abundances of the community.
`Nexp`	A numeric vector containing species abundances of the metacommunity.
`NorP`	A numeric vector, an integer vector, an abundance vector (`AbdVector`) or a probability vector (`ProbaVector`). Contains either abundances or probabilities of the community.
`NorPexp`	A numeric vector, an integer vector, an abundance vector (`AbdVector`) or a probability vector (`ProbaVector`). Contains either abundances or probabilities of the metacommunity.
`Correction`	A string containing one of the possible corrections: currently, `"ChaoShen"` (Marcon *et al.*, 2012) equivalent to `"Best"`, and `"ZhangGrabchak"` (Zhang and Grabchak, 2014).
`...`	Additional arguments. Unused.
`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

The derivation of Shannon beta entropy can be found in Marcon *et al.* (2012).

Bias correction requires the number of individuals to estimate sample `Coverage`. Use `bcShannonBeta` and choose the `Correction`.

The functions are designed to be used as simply as possible. `ShannonBeta` is a generic method. If its first argument is an abundance vector, an integer vector or a numeric vector which does not sum to 1, the bias corrected function `bcShannonBeta` is called. Explicit calls to `bcShannonBeta` (with bias correction) or to `ShannonBeta.ProbaVector` (without correction) are possible to avoid ambiguity. The `.integer` and `.numeric` methods accept `Ps` or `Ns` arguments instead of `NorP` for backward compatibility.

Value

A number equal to the calculated entropy.

References

Marcon, E., Herault, B., Baraloto, C. and Lang, G. (2012). The Decomposition of Shannon's Entropy and a Confidence Interval for Beta Diversity. *Oikos* 121(4): 516-522.

Zhang, Z. and Grabchak M. (2014). Nonparametric Estimation of Kullback-Leibler Divergence. *Neural computation* 26(11): 2570-2593.

See Also

`bcShannonBeta`

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest)
data(Paracou618)
# Ps is the vector of probabilities
Ps <- as.ProbaVector(Paracou618.MC$Ps)
# Probability distribution of the first plot
Ps1 <- as.ProbaVector(Paracou618.MC$Psi[, 1])
# Shannon beta entropy of the plot
ShannonBeta(Ps1, Ps)

# Ns is the vector of abundances of the metacommunity
Ns <- as.AbdVector(Paracou618.MC$Ns)
# Abundances in the first plot
Ns1 <- as.AbdVector(Paracou618.MC$Nsi[, 1])
# Reduced-bias estimator of Shannon beta entropy of the plot
bcShannonBeta(Ns1, Ns)
```

`Simpson` *Simpson entropy of a community*

Description

Calculates the Simpson entropy of a probability vector.

Usage

```
Simpson(NorP, ...)
bcSimpson(Ns, Correction = "Best", CheckArguments = TRUE)
## S3 method for class 'ProbaVector'
Simpson(NorP, ..., CheckArguments = TRUE,
  Ps = NULL)
## S3 method for class 'AbdVector'
Simpson(NorP, Correction="Best", Level = NULL, ...,
  CheckArguments = TRUE, Ns = NULL)
## S3 method for class 'integer'
Simpson(NorP, Correction="Best", Level = NULL, ...,
  CheckArguments = TRUE, Ns = NULL)
```

```
## S3 method for class 'numeric'
Simpson(NorP, Correction="Best", Level = NULL, ...,
  CheckArguments = TRUE, Ps = NULL, Ns = NULL)
```

Arguments

`Ps`	A probability vector, summing to 1.
`Ns`	A numeric vector containing species abundances.
`NorP`	A numeric vector, an integer vector, an abundance vector (`AbdVector`) or a probability vector (`ProbaVector`). Contains either abundances or probabilities.
`Correction`	A string containing one of the possible corrections accepted by `bcTsallis` or `"Lande"`. `"Best"`, the default value, is currently `"Jackknife"`. Ignored by interpolation and extrapolation.
`Level`	The level of interpolation or extrapolation. It may be an a chosen sample size (an integer) or a sample coverage (a number between 0 and 1).
`...`	Additional arguments. Unused.
`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

Lande's correction has been derived (Lande, 1996; Good, 1953) especially for Simpson entropy, while other corrections are for generalized Tsallis entropy. It is identical to the unbiased estimator proposed by Simpson, although arguments were different. It equals the plug-in etimator multiplied by n/(n-1) where n is the total number of individuals.

Bias correction requires the number of individuals to estimate sample `Coverage`.

The functions are designed to be used as simply as possible. `Simpson` is a generic method. If its first argument is an abundance vector, an integer vector or a numeric vector which does not sum to 1, the bias corrected function `bcSimpson` is called.

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. Simpson's extrapolated entropy estimator does not rely on the estimation of the asymptotic distribution.

Value

A named number equal to the calculated entropy. The name is that of the bias correction used.

References

Chao, A., Gotelli, N. J., Hsieh, T. C., Sander, E. L., Ma, K. H., Colwell, R. K., Ellison, A. M (2014). Rarefaction and extrapolation with Hill numbers: A framework for sampling and estimation in species diversity studies. *Ecological Monographs*, 84(1): 45-67.

Good, I. J. (1953). On the Population Frequency of Species and the Estimation of Population Parameters. *Biometrika* 40(3/4): 237-264.

Lande, R. (1996). Statistics and partitioning of species diversity, and similarity among multiple communities. *Oikos* 76: 5-13.

Simpson, E. H. (1949). Measurement of diversity. *Nature* 163(4148): 688.

See Also

Tsallis, bcSimpson

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest)
data(Paracou618)
# Ns is the total number of trees per species
Ns <- as.AbdVector(Paracou618.MC$Ns)
# Whittaker plot
plot(Ns)

# Calculate an unbiased estimator of Simpson's index of diversity
Simpson(Ns)
```

`SimpsonBeta` *Simpson beta entropy of a community*

Description

Calculates the Simpson beta entropy of a community belonging to a metacommunity.

Usage

```
SimpsonBeta(NorP, NorPexp = NULL, ...)
bcSimpsonBeta(Ns, Nexp, Correction = "Best", CheckArguments = TRUE)
## S3 method for class 'ProbaVector'
SimpsonBeta(NorP, NorPexp = NULL, ...,
  CheckArguments = TRUE, Ps = NULL, Pexp = NULL)
## S3 method for class 'AbdVector'
SimpsonBeta(NorP, NorPexp = NULL, Correction = "Best", ...,
  CheckArguments = TRUE, Ns = NULL, Nexp = NULL)
## S3 method for class 'integer'
SimpsonBeta(NorP, NorPexp = NULL, Correction = "Best", ...,
  CheckArguments = TRUE, Ns = NULL, Nexp = NULL)
## S3 method for class 'numeric'
SimpsonBeta(NorP, NorPexp = NULL, Correction = "Best", ...,
  CheckArguments = TRUE, Ps = NULL, Ns = NULL, Pexp = NULL, Nexp = NULL)
```

Arguments

`Ps`	The probability vector of species of the community.
`Pexp`	The probability vector of species of the metacommunity.
`Ns`	A numeric vector containing species abundances of the community.
`Nexp`	A numeric vector containing species abundances of the metacommunity.

`NorP`	A numeric vector, an integer vector, an abundance vector (`AbdVector`) or a probability vector (`ProbaVector`). Contains either abundances or probabilities of the community.
`NorPexp`	A numeric vector, an integer vector, an abundance vector (`AbdVector`) or a probability vector (`ProbaVector`). Contains either abundances or probabilities of the metacommunity.
`Correction`	A string containing one of the possible corrections: currently, only `"ChaoShen"`, identical to `"Best"`.
`...`	Additional arguments. Unused.
`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

The derivation of Tsallis beta entropy (Simpson is Tsallis of order 2) can be found in Marcon et al. (2014).

Bias correction requires the number of individuals to estimate sample `Coverage`. Use `bcSimpsonBeta` and choose the `Correction`.

Note that Simpson beta entropy value is related to Simpson alpha entropy value and cannot be compared accross communities (Jost, 2007). Beta entropy of a community is not meaningful in general, do rather calculate the `BetaDiversity` of order 2 of the metacommunity.

The functions are designed to be used as simply as possible. `SimpsonBeta` is a generic method. If its first argument is an abundance vector, an integer vector or a numeric vector which does not sum to 1, the bias corrected function `bcSimpsonBeta` is called. Explicit calls to `bcSimpsonBeta` (with bias correction) or to `SimpsonBeta.ProbaVector` (without correction) are possible to avoid ambiguity. The `.integer` and `.numeric` methods accept `Ps` or `Ns` arguments instead of `NorP` for backward compatibility.

Value

A named number equal to the calculated entropy. The name is that of the bias correction used.

References

Jost (2007), Partitioning diversity into independent alpha and beta components. *Ecology* 88(10): 2427-2439.

Marcon, E., Scotti, I., Herault, B., Rossi, V. and Lang, G. (2014). Generalization of the partitioning of Shannon diversity. *PLOS One* 9(3): e90289.

See Also

`Simpson`, `bcSimpsonBeta`, `BetaDiversity`

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest)
data(Paracou618)
# Ps is the vector of probabilities
Ps <- as.ProbaVector(Paracou618.MC$Ps)
# Probability distribution of the first plot
Ps1 <- as.ProbaVector(Paracou618.MC$Psi[, 1])

# Simpson beta entropy of the plot
SimpsonBeta(Ps1, Ps)
# Transform into diversity
expq(SimpsonBeta(Ps1, Ps)/(1-Simpson(Ps1)), 2)

# Ns is the vector of abundances of the metacommunity
Ns <- as.AbdVector(Paracou618.MC$Ns)
# Abundances in the first plot
Ns1 <- as.AbdVector(Paracou618.MC$Nsi[, 1])
# Reduced-bias Shannon beta entropy of the plot
bcSimpsonBeta(Ns1, Ns)
```

`SimTest` *SimTest class*

Description

Methods for objects of type "`SimTest`", used to test a value against its distribution under a simulated null hypothesis.

Usage

```
as.SimTest(RealValue, SimulatedValues)
is.SimTest(x)
## S3 method for class 'SimTest'
autoplot(object, Quantiles = c(0.025, 0.975), ...,
  colValue = "red", colQuantiles = "black", ltyQuantiles = 2,
  main = NULL, xlab = "Simulated Values", ylab = "Density")
## S3 method for class 'SimTest'
plot(x, Quantiles = c(0.025, 0.975), ...,
  colValue = "red", lwdValue = 2, ltyValue = 2,
  colQuantiles = "black", lwdQuantiles = 1, ltyQuantiles = 2,
  main = NULL, xlab = "Simulated Values", ylab = "Density")
## S3 method for class 'SimTest'
summary(object, Quantiles = c(0.025, 0.975), ...)
```

Arguments

`x`	An object to be tested or plotted.
`object`	An object.

`RealValue`	A numeric Value (the actual one).
`SimulatedValues`	A numeric vector containing the simulated values.
`Quantiles`	A vector containing the quantiles of interest.
`colValue`	The color of the line representing the real value on the plot.
`lwdValue`	The width of the line representing the real value on the plot.
`ltyValue`	The line type of the line representing the real value on the plot.
`colQuantiles`	The color of the lines representing the quantiles on the plot.
`lwdQuantiles`	The width of the lines representing the quantiles on the plot.
`ltyQuantiles`	The line type of the lines representing the quantiles on the plot.
`main`	The main title of the plot. if `NULL` (by default), there is no title.
`xlab`	The X axis label.
`ylab`	The Y axis label.
`...`	Additional arguments to be passed to the generic methods.

Details

Simulated values should be obtained by simulation. The actual value is compared to simulated quantiles. `SimTest` objects can be plotted and summarized.

Value

`SimTest` objects are lists containing:

`RealValue`	The value to test.
`SimulatedValues`	A vector of simulated values, whose quantiles will be used for the test.

`is.SimTest` returns `TRUE` if the object is of class `SimTest`.

`summary.SimTest` returns a summary of the object, including the empirical quantile of the real value in the simulated distributon.

Examples

```
# Set the value to test
Real <- 0.8
# Is it a realization of a Gaussian distribution?
Sims <- rnorm(1000)
# Make a Simtest object
st <- as.SimTest(Real, Sims)
summary(st)
# Plot
plot(st)
# ggplot
autoplot(st)
```

`SpeciesDistribution` *Species Distributions*

Description

A Species Distribution is a (preferably named) vector containing species abundances or probabilities.

Usage

```
as.SpeciesDistribution(x, ...)
## S3 method for class 'data.frame'
as.SpeciesDistribution(x, ...)
## S3 method for class 'integer'
as.SpeciesDistribution(x, ...)
## S3 method for class 'numeric'
as.SpeciesDistribution(x, ...)
## S3 method for class 'SpeciesDistribution'
autoplot(object, ..., Distribution = NULL,
         ylog = TRUE, main = NULL, xlab = "Rank", ylab = NULL,
         pch = 19,
         col = "black",
         cex = 1.5)
## S3 method for class 'SpeciesDistribution'
plot(x, ..., Distribution = NULL,
         type = "b", log = "y", main = NULL, xlab = "Rank", ylab = NULL)
is.SpeciesDistribution(x)
as.ProbaVector(x, ...)
## S3 method for class 'data.frame'
as.ProbaVector(x, ...)
## S3 method for class 'integer'
as.ProbaVector(x, Correction = "None", Unveiling = "None",
          RCorrection = "Jackknife", JackOver = FALSE, JackMax = 10,
          CEstimator = "ZhangHuang", q = 0, ..., CheckArguments = TRUE)
## S3 method for class 'numeric'
as.ProbaVector(x, Correction = "None", Unveiling = "None",
          RCorrection = "Jackknife", JackOver = FALSE, JackMax = 10,
          CEstimator = "ZhangHuang", q = 0, ..., CheckArguments = TRUE)
is.ProbaVector(x)
as.AbdVector(x, ...)
## S3 method for class 'data.frame'
as.AbdVector(x, Round = TRUE, ...)
## S3 method for class 'integer'
as.AbdVector(x, ...)
## S3 method for class 'numeric'
as.AbdVector(x, Round = TRUE, ...)
is.AbdVector(x)
```

Arguments

`x`	An object.
`object`	An object.
`Distribution`	The distribution to fit on the plot. May be `"lnorm"` (log-normal), `"lseries"` (log-series), `"geom"` (geometric) or `"bstick"` (broken stick). If `NULL`, no distribution is fitted. See `rCommunity` for the description of these distributions.
`Round`	If `TRUE` (by default), values of `x` are set to integer to create an `AbdVector`. This is useful if original abundances are not integers (this is often the case for `MetaCommunity` abundances which are the product of probabilities by the number of individuals) and integer values are required (for example to calculate the bootstrap confidence interval of a community profile).
`Correction`	A string containing one of the possible corrections to estimate a probability distribution: `"None"` (no correction, the default value), or `"Chao2013"`, `"Chao2015"`, `"ChaoShen"` to estimate the probability of the observed species in the asymptotic distribution.
`Unveiling`	A string containing one of the possible unveiling methods to estimate the probabilities of the unobserved species: `"None"` (default, no species is added), `"unif"` (uniform: all unobserved species have the same probability) or `"geom"` (geometric: the unobserved species distribution is geometric).
`RCorrection`	A string containing a correction recognized by `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 order `q` of the asymptotic distribution rarefied to the observed sample size equals the actual entropy of the data.
`JackOver`	If `TRUE`, retain the jackknife order immediately superior to the optimal one, usually resulting in the overestimation of the number of species. Default is `FALSE`. Ignored if `RCorrection` is not `"Jackknife"`.
`JackMax`	The highest jackknife order allowed. Default is 10. Allowed values are between 1 and 10.
`CEstimator`	A string containing an estimator recognized by `Coverage` to evaluate the sample coverage. `"ZhangHuang"` is the default value.
`q`	A number: the order of entropy. Default is 0 for richness. Used only to estimate asymptotic probability distributions with `RCorrection` equal to `"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.
`type`	The plot type, see `plot`.
`log`	The axis to plot in log scale, *e.g.* `"xy"` for both axes. Default is `"y"`.
`main`	The main title of the plot. if `NULL` (by default), there is no title.
`xlab`	The X axis label, "Rank" by default.
`ylab`	The Y axis label. if `NULL` (by default), "Probability" or "Abundance" is chosen according to the object class.
`ylog`	Logical; if `TRUE` (by default), the Y-axis of the plot is log scaled.

`pch`	The plotting characters. See `points`.
`col`	The color of the geom objects. See "Color Specification" in `par`.
`cex`	The character expansion (size) of the points. See `points`.
`...`	Additional arguments to be passed to `plot`. Unused elsewhere.
`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

`SpeciesDistribution` objects include `AbdVector` and `ProbaVector` objects.

`as.AbdVector` just sets the class of the numeric or integer `x` so that appropriate versions of community functions (generic methods such as `Diversity`) are applied. Abundance values are rounded (by default) to the nearest integer.

`as.ProbaVector` normalizes the vector so that it sums to 1. If `Correction` is not `"None"`, the observed abundance distribution is used to estimate the actual species distribution. The list of species will be changed: zero-abundance species will be cleared, and some unobserved species will be added. First, observed species probabilities are estimated folllowing Chao and Shen (2003), *i.e.* input probabilities are multiplied by the sample coverage, or according to more sophisticated models: Chao *et al.* (2013, single-parameter model), or Chao *et al.* (2015, 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 `RCorrection` (if the Jackknife estimator is chosen, the `JackOver` argument allows using the order immediately over the optimal one). The coverage deficit (1 minus the sample coverage) is shared by the unobserved species equally (`Unveiling = "unif"`, Chao *et al.*, 2013) or according to a geometric distribution (`Unveiling = "geom"`, Chao *et al.*, 2015).

These functions can be applied to data frames to calculate the joint diversity (Gregorius, 2010).

`SpeciesDistribution` objects can be plotted. The `plot` method returns the estimated parameters of the fitted distribution. The broken stick has no parameter, so the maximum abundance is returned.

Note

Fisher's alpha (Fisher *et al.*, 1943) is estimated to fit the log-series distribution. The estimation is done by the `fisher.alpha` function of package `vegan`. It may differ substantially from the estimation returned by `optimal.theta` from package `untb`.

Author(s)

Eric Marcon <Eric.Marcon@agroparistech.fr>, Bruno Herault <Bruno.Herault@cirad.fr>

References

Chao, A. and Shen, T. J. (2003). Nonparametric estimation of Shannon's index of diversity when there are unseen species in sample. *Environmental and Ecological Statistics* 10(4): 429-443.

Chao, A., Wang, Y. T. and 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.

Chao, A., Hsieh, T. C., Chazdon, R. L., Colwell, R. K., Gotelli, N. J. (2015) Unveiling the Species-Rank Abundance Distribution by Generalizing Good-Turing Sample Coverage Theory. *Ecology* 96(5): 1189-1201.

Fisher R.A., Corbet A.S., Williams C.B. (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.

Gregorius H.-R. (2010) Linking Diversity and Differentiation. *Diversity* 2(3): 370-394.

See Also

`rgeom`, `rlnorm`, `rCommunity`, `RAClnorm`

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest)
data(Paracou618)
# Ns is the total number of trees per species
Ns <- as.AbdVector(Paracou618.MC$Ns)
# Whittaker plot, poorly fitted by a log-normal distribution
plot(Ns, Distribution = "lnorm")
# ggplot version
autoplot(Ns, Distribution = "lnorm")
```

Tsallis *Tsallis (HCDT) Entropy of a community*

Description

Calculates the HCDT, also known as Tsallis entropy of order q of a probability vector.

Usage

```
Tsallis(NorP, q = 1, ...)
bcTsallis(Ns, q = 1, Correction = "Best", SampleCoverage = NULL,
  CheckArguments = TRUE)
## S3 method for class 'ProbaVector'
Tsallis(NorP, q = 1, ...,
  CheckArguments = TRUE, Ps = NULL)
## S3 method for class 'AbdVector'
Tsallis(NorP, q = 1, Correction = "Best", Level = NULL,
  PCorrection="Chao2015", Unveiling="geom", RCorrection="Rarefy", ...,
  CheckArguments = TRUE, Ns = NULL)
## S3 method for class 'integer'
Tsallis(NorP, q = 1, Correction = "Best", Level = NULL,
  PCorrection="Chao2015", Unveiling="geom", RCorrection="Rarefy", ...,
  CheckArguments = TRUE, Ns = NULL)
## S3 method for class 'numeric'
```

```
Tsallis(NorP, q = 1, Correction = "Best", Level = NULL,
  PCorrection="Chao2015", Unveiling="geom", RCorrection="Rarefy", ...,
  CheckArguments = TRUE, Ps = NULL, Ns = NULL)
```

Arguments

`Ps`	A probability vector, summing to 1.
`Ns`	A numeric vector containing species abundances.
`NorP`	A numeric vector, an integer vector, an abundance vector (`AbdVector`) or a probability vector (`ProbaVector`). Contains either abundances or probabilities.
`q`	A number: the order of entropy. Some corrections allow only a positive number. Default is 1 for Shannon entropy.
`Correction`	A string containing one of the possible asymptotic estimators: `"None"` (no correction), `"ChaoShen"`, `"GenCov"`, `"Grassberger"`, `"Holste"`, `"Bonachela"`, `"ZhangGrabchak"`, or `"ChaoJost"`, `"Marcon"`, `"UnveilC"`, `"UnveiliC"`, `"UnveilJ"` or `"Best"`, the default value. Currently, `"Best"` is `"UnveilJ"`.
`Level`	The level of interpolation or extrapolation. It may be an a chosen sample size (an integer) or a sample coverage (a number between 0 and 1).
`PCorrection`	A string containing one of the possible corrections to estimate a probability distribution in `as.ProbaVector`: `"Chao2015"` is the default value. Used only for extrapolation.
`Unveiling`	A string containing one of the possible unveiling methods to estimate the probabilities of the unobserved species in `as.ProbaVector`: `"geom"` (the unobserved species distribution is geometric) is the default value. Used only for extrapolation.
`RCorrection`	A string containing a correction recognized by `Richness` to evaluate the total number of species in `as.ProbaVector`. `"Rarefy"` is the default value to estimate the number of species such that the entropy of the asymptotic distribution rarefied to the observed sample size equals the observed entropy of the data. Used only for extrapolation.
`SampleCoverage`	The sample coverage of `Ns` calculated elsewhere. Used to calculate the gamma diversity of meta-communities, see details.
`...`	Additional arguments. Unused.
`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

Tsallis (Havrda and Charvat, 1967; Daroczy, 1970; Tsallis, 1988) generalized entropy is a generalized measure of diversity (Jost, 2006).

Bias correction requires the number of individuals to estimate sample `Coverage`.

Correction 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 and Jost (2015) and Marcon (2015).

The `"ChaoJost"` (Chao, Wang and Jost, 2013 for $q = 1$; Chao and Jost, 2015) estimator 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 very slow if the number of individuals is more than a few hundreds. This estimator was named "ChaoWangJost" in previous versions of the package; its old name is still supported for backward compatibility.

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 jacknife (`"UnveilJ"`). The `"UnveilJ"` correction often has a lower bias but a greater variance (Marcon, 2015). It is a good first choice thanks to the versatility of the jacknife estimator of richness.

The functions are designed to be used as simply as possible. `Tsallis` is a generic method. If its first argument is an abundance vector, an integer vector or a numeric vector which does not sum to 1, the bias corrected function `bcTsallis` is called.

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 `SampleCoverage` argument allows applying the `"ChaoShen"` and `"Grassberger"` corrections to estimate quite well the entropy. `DivPart` and `GammaEntropy` functions use this tweak.

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. Special cases q equals 0, 1 or 2 are treated by `Richness`, `Shannon` and `Simpson` functions. For extrapolation of entropy of other values of q, the asymptotic distribution of the community must be estimated by `as.ProbaVector`. The default arguments allow joining smoothly the extrapolated entropy and the observed entropy by estimating the number of unobserved species so that the entropy of the observed distribution equals the entropy of the asymptotic distribution rarefied to the actual sample size.

Value

A named number equal to the calculated entropy. The name is that of the bias correction used.

References

Chao, A., Gotelli, N. J., Hsieh, T. C., Sander, E. L., Ma, K. H., Colwell, R. K., Ellison, A. M (2014). Rarefaction and extrapolation with Hill numbers: A framework for sampling and estimation in species diversity studies. *Ecological Monographs*, 84(1): 45-67.

Chao, A. and Jost, L. (2015) Estimating diversity and entropy profiles via discovery rates of new species. *Methods in Ecology and Evolution* 6(8): 873-882.

Chao, A., Hsieh, T. C., Chazdon, R. L., Colwell, R. K., Gotelli, N. J. (2015) Unveiling the Species-Rank Abundance Distribution by Generalizing Good-Turing Sample Coverage Theory. *Ecology* 96(5): 1189-1201.

Chao, A., Wang, Y. T. and 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.

Havrda, J. and Charvat, F. (1967). Quantification method of classification processes. Concept of structural a-entropy. *Kybernetika* 3(1): 30-35.

Daroczy, Z. (1970). Generalized information functions. *Information and Control* 16(1): 36-51.

Jost, L. (2006). Entropy and diversity. *Oikos* 113(2): 363-375.

Marcon, E. (2015) Practical Estimation of Diversity from Abundance Data. *HAL* 01212435: 1-27.

Marcon, E., Scotti, I., Herault, B., Rossi, V. and Lang, G. (2014). Generalization of the partitioning of Shannon diversity. *PLOS One* 9(3): e90289.

Tsallis, C. (1988). Possible generalization of Boltzmann-Gibbs statistics. *Journal of Statistical Physics* 52(1): 479-487.

Zhang, Z., and Grabchak, M. (2016). Entropic Representation and Estimation of Diversity Indices. *Journal of Nonparametric Statistics*, 28(3): 563-575.

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest)
data(Paracou618)
# Ns is the total number of trees per species
Ns <- as.AbdVector(Paracou618.MC$Ns)
# Species probabilities
Ps <- as.ProbaVector(Paracou618.MC$Ns)
# Whittaker plot
plot(Ns)

# Calculate entropy of order 1, i.e. Shannon's entropy
Tsallis(Ps, 1)
# Calculate it with estimation bias correction
Tsallis(Ns, 1)
```

`TsallisBeta` *Tsallis beta entropy of a community*

Description

Calculates the Tsallis beta entropy of order q of a community belonging to a metacommunity.

Usage

```
TsallisBeta(NorP, NorPexp = NULL, q = 1, ...)
bcTsallisBeta(Ns, Nexp = NULL, q, Correction = "Best", CheckArguments = TRUE)
## S3 method for class 'ProbaVector'
TsallisBeta(NorP, NorPexp = NULL, q = 1, ...,
  CheckArguments = TRUE, Ps = NULL, Pexp = NULL)
## S3 method for class 'AbdVector'
TsallisBeta(NorP, NorPexp = NULL, q = 1, Correction = "Best", ...,
  CheckArguments = TRUE, Ns = NULL, Nexp = NULL)
## S3 method for class 'integer'
```

```
TsallisBeta(NorP, NorPexp = NULL, q = 1, Correction = "Best", ...,
  CheckArguments = TRUE, Ns = NULL, Nexp = NULL)
## S3 method for class 'numeric'
TsallisBeta(NorP, NorPexp = NULL, q = 1, Correction = "Best", ...,
  CheckArguments = TRUE, Ps = NULL, Ns = NULL, Pexp = NULL, Nexp = NULL)
```

Arguments

`Ps`	The probability vector of species of the community.
`Pexp`	The probability vector of species of the metacommunity.
`Ns`	A numeric vector containing species abundances of the community.
`Nexp`	A numeric vector containing species abundances of the metacommunity.
`NorP`	A numeric vector, an integer vector, an abundance vector (`AbdVector`) or a probability vector (`ProbaVector`). Contains either abundances or probabilities of the community.
`NorPexp`	A numeric vector, an integer vector, an abundance vector (`AbdVector`) or a probability vector (`ProbaVector`). Contains either abundances or probabilities of the metacommunity.
`q`	A number: the order of entropy. Default is 1 for Shannon entropy.
`Correction`	A string containing one of the possible corrections: currently, only `"ChaoShen"` or `"None"`. `"Best"` is the default value, it is equivalent to `"ChaoShen"`.
`...`	Additional arguments. Unused.
`CheckArguments`	Logical; if `TRUE`, the function arguments are verified. Should be set to `FALSE` to save time when the arguments have been checked elsewhere.

Details

The derivation of Tsallis beta entropy can be found in Marcon et al. (2014).

Bias correction requires the number of individuals to estimate sample `Coverage`. Use `bcTsallisBeta` and choose the `Correction`.

Note that beta entropy value is related to alpha entropy (if q is not 1) and cannot be compared accross communities (Jost, 2007). Beta entropy of a community is not meaningful in general, do rather calculate the `BetaDiversity` of the metacommunity.

The functions are designed to be used as simply as possible. `TsallisBeta` is a generic method. If its first argument is an abundance vector, an integer vector or a numeric vector which does not sum to 1, the bias corrected function `bcTsallisBeta` is called. Explicit calls to `bcTsallisBeta` (with bias correction) or to `TsallisBeta.ProbaVector` (without correction) are possible to avoid ambiguity. The `.integer` and `.numeric` methods accept `Ps` or `Ns` arguments instead of `NorP` for backward compatibility.

Value

A named number equal to the calculated entropy. The name is that of the bias correction used.

References

Jost (2007), Partitioning diversity into independent alpha and beta components. *Ecology* 88(10): 2427-2439.

Marcon, E., Scotti, I., Herault, B., Rossi, V. and Lang, G. (2014). Generalization of the partitioning of Shannon diversity. *PLOS One* 9(3): e90289.

Examples

```
# Load Paracou data (number of trees per species in two 1-ha plot of a tropical forest)
data(Paracou618)
# Ps is the vector of probabilities
Ps <- Paracou618.MC$Ps
# Probability distribution of the first plot
Ps1 <- Paracou618.MC$Psi[, 1]
# Divergence of order 2 between plot 1 and the whole forest
TsallisBeta(Ps1, Ps, 2)

# Ns is the vector of abundances of the metacommunity
Ns <- Paracou618.MC$Ns
# Abundances in the first plot
Ns1 <- Paracou618.MC$Nsi[, 1]
# Divergence of order 2 between plot 1 and the whole forest, with bias correction
bcTsallisBeta(Ns1, Ns, 2)
```

Index