Package 'rdiversity'

May 20, 2020

Type Package

Title Measurement and Partitioning of Similarity-Sensitive Biodiversity

Version 2.0

Date 2020-05-05

URL https://github.com/boydorr/rdiversity

BugReports https://github.com/boydorr/rdiversity/issues

Description Provides a framework for the measurement and partitioning of the (similarity-sensitive) biodiversity of a metacommunity and its constituent subcommunities. Richard Reeve, et al. (2016) <arXiv:1404.6520v3>.

License GPL-3

Depends R (>= 2.10)

Imports binaryLogic, methods, reshape2, stats, utils

Suggests ape, testthat, knitr, rmarkdown, covr, vcfR

LazyData true

RoxygenNote 7.1.0

Collate 'chainsaw.R' 'check_partition.R' 'check_phypartition.R' 'check_similarity.R' 'class-distance.R' 'class-metacommunity.R' 'class-powermean.R' 'class-relativeentropy.R' 'class-similarity.R' 'similarity.R' 'dist2sim.R' 'distance.R' 'metadiv.R' 'subdiv.R' 'metacommunity.R' 'diversity-components.R' 'diversity-measures.R' 'gen2dist.r' 'geneid.R' 'genevec.R' 'hs_parameters.R' 'inddiv.R' 'phy2branch.R' 'phy2dist.R' 'phy_abundance.R' 'phy_struct.R' 'power_mean.R' 'powermean.R' 'rdiversity-package.R' 'relativeentropy.R' 'repartition.R' 'smatrix.R' 'summarise.R' 'tax2dist.R' 'taxfac.R' 'taxid.R' 'taxmask.R' 'taxvec.R' 'tbar.R' 'zmatrix.R'

Encoding UTF-8

NeedsCompilation no

Author Sonia Mitchell [cre, aut] (<https://orcid.org/0000-0003-1536-2066>),
Richard Reeve [aut, ths] (<https://orcid.org/0000-0003-2589-8091>),
Tom White [ctb] (<https://orcid.org/0000-0002-9639-3800>)

Maintainer Sonia Mitchell <sonia.mitchell@glasgow.ac.uk>

Repository CRAN

Date/Publication 2020-05-20 13:10:05 UTC

R topics documented:

`rdiversity-package` *rdiversity: diversity measurement in R*

Description

`rdiversity` is an R package based around a framework for measuring and partitioning biodiversity using similarity-sensitive diversity measures. It provides functionality for measuring alpha, beta and gamma diversity of metacommunities (*e.g.* ecosystems) and their constituent subcommunities, where similarity may be defined as taxonomic, phenotypic, genetic, phylogenetic, functional, and so on. It uses the diversity measures described in the arXiv paper, '*How to partition diversity*'.

Details

- For more information go to our GitHub page; `https://github.com/boydorr/rdiversity`
- Please raise an issue if you find any problems; `https://github.com/boydorr/rdiversity/issues`
- This package is cross-validated against our Julia package; `https://github.com/richardreeve/Diversity.jl`

Author(s)

Sonia Mitchell <sonia.mitchell@glasgow.ac.uk> (maintainer)
Richard Reeve <richard.reeve@glasgow.ac.uk>

References

Reeve, R., T. Leinster, C. Cobbold, J. Thompson, N. Brummitt, S. Mitchell, and L. Matthews. 2016. How to partition diversity. (https://arxiv.org/abs/1404.6520)

`ancestral_nodes` *ancestral_nodes*

Description

ancestral_nodes

Usage

```
ancestral_nodes(tree, node)
```

Arguments

`tree`	object of class `phylo`.
`node`	object of class `numeric`.

`chainsaw` *Function to cut the phylogeny to a specified depth from the tip with the greatest distance from the root.*

Description

Function to cut the phylogeny to a specified depth from the tip with the greatest distance from the root.

Usage

```
chainsaw(partition, ps, depth)
```

Arguments

partition	two-dimensional matrix of mode numeric with rows as types (species), columns as subcommunities, and each element containing the relative abundance of types in each subcommunity relative to the metacommunity as a whole. In the phylogenetic case, this corresponds to the proportional abundance of historical species, which is calculated from the proportional abundance of terminal taxa
ps	phy_struct() output
depth	proportion of total tree height to be conserved (taken as a proportion from the highest tip). Describes how far back we go in the tree, with 0 marking the date of the most recent tip, and 1 marking the most recent common ancestor. Numbers greater than 1 extend the root of the tree

Value

chainsaw() returns an object of class metacommunity

---

check_partition *Check partition matrix*

Description

check_partition() is used to validate partition matrices.

Usage

```
check_partition(partition)
```

Arguments

partition	two-dimensional matrix of mode numeric with rows as types (species), columns as subcommunities, and each element containing the relative abundance of types in each subcommunity relative to the metacommunity as a whole. In the phylogenetic case, this corresponds to the proportional abundance of historical species, which is calculated from the proportional abundance of terminal taxa

Value

Returns a two-dimensions matrix of mode numeric. If the partition matrix was valid, this should be identical to that which was input as an argument.

check_phypartition *check_phypartition*

Description

check_phypartition() is used to validate partition matrices for use with phylogenies.

Usage

```
check_phypartition(tip_labels, partition)
```

Arguments

tip_labels	vector containing elements of class character.
partition	two-dimensional matrix of mode numeric with rows as types (species), columns as subcommunities, and each element containing the relative abundance of types in each subcommunity relative to the metacommunity as a whole. In the phylogenetic case, this corresponds to the proportional abundance of historical species, which is calculated from the proportional abundance of terminal taxa

Value

Returns a two-dimensions matrix of mode numeric. If the partition matrix was valid, this should be identical to that which was input as an argument.

check_similarity *Check similarity matrix*

Description

check_similarity() is used to validate similarity matrices.

Usage

```
check_similarity(similarity, partition)
```

Arguments

similarity	two-dimensional matrix of mode numeric; contains pair-wise similarity between types.
partition	two-dimensional matrix of mode numeric with rows as types (species), columns as subcommunities, and each element containing the relative abundance of types in each subcommunity relative to the metacommunity as a whole. In the phylogenetic case, this corresponds to the proportional abundance of historical species, which is calculated from the proportional abundance of terminal taxa

Value

Returns a two-dimensions `matrix` of mode `numeric`. If the similarity matrix was valid, this should be identical to that which was input as an argument.

`descendant_tips` *descendant_tips*

Description

descendant_tips

Usage

```
descendant_tips(tree, node)
```

Arguments

`tree`	object of class `phylo`.
`node`	object of class `numeric`.

`dist2sim` *Distance to similarity*

Description

Converts `distance` objects into `similarity` objects.

Usage

```
dist2sim(dist, transform, k = 1, normalise = TRUE, max_d)
```

Arguments

`dist`	object of class `distance`
`transform`	object of class `character`, can be either "linear" or "exponential"
`k`	scaling parameter
`normalise`	object of class `logical`, which when TRUE will normalise distances to one
`max_d`	object of class `numeric`

Details

Distances can be transformed either *linearly* or *exponentially*. That is `1 -k * dist` for non-negative values, or `exp(-k * dist)`, respectively. If `normalise` is true, then `dist = dist/max_d`.

Value

`dist2sim(x)` returns an object of class `similarity`.

distance *Generate distance object*

Description

Container for class distance.

Usage

```
distance(distance, dat_id)

## S4 method for signature 'matrix,character'
distance(distance, dat_id)

## S4 method for signature 'matrix,missing'
distance(distance, dat_id)
```

Arguments

distance	distance matrix
dat_id	object of class character denoting the type of diversity being calculated. This can be "naive", "genetic", "taxonomic", and so on

Value

distance() returns an object of class distance.

distance-class *distance-class*

Description

Container for class distance.

Usage

```
## S4 method for signature 'distance'
show(object)
```

Arguments

object	object of class distance

Fields

`distance` two-dimensional `matrix` of mode `numeric` with rows as types, columns as types, and elements containing the pairwise distance of types

`dat_id` object of class `character` describing the class of distance / similarity being used, e.g. "naive", "taxonomic", and so on

`components` list containing the components necessary to calculate similarity. This list is empty when `precompute_dist = TRUE` when calculating distance. When a pairwise distance matrix is too large and `precompute_dist = FALSE`, this list contains all the information required to calculate pairwise distance between types

---

gen2dist *Genetic distance matrix*

Description

Converts a vcfR object to a matrix of pairwise genetic distances.

Usage

```
gen2dist(vcf)
```

Arguments

`vcf`	object of class `vcfR`.

Value

`gen2dist(x)` returns an object of class `distance` containing a `matrix` of pairwise genetic distances.

---

geneid *geneid*

Description

Converts a single sequence

Usage

```
geneid(seq, kmer = 16)
```

Arguments

`seq`	a single read
`kmer`	is a 16-mer by default

genevec *genevec*

Description

genevec

Usage

```
genevec(one, two)
```

Arguments

one	Sequence one
two	Sequence two

hs_parameters *Historical species parameters*

Description

Internal function, which extracts various parameters associated with historical species.

Usage

```
hs_parameters(tree)
```

Arguments

tree	object of class phylo.

Value

Returns parameters associated with each historic species.

inddiv *Calculate individual-level diversity*

Description

Generic function for calculating individual-level diversity.

Usage

```
inddiv(data, qs)

## S4 method for signature 'powermean'
inddiv(data, qs)

## S4 method for signature 'relativeentropy'
inddiv(data, qs)

## S4 method for signature 'metacommunity'
inddiv(data, qs)
```

Arguments

data	matrix of mode numeric; containing diversity components
qs	vector of mode numeric containing q values

Details

data may be input as three different classes:

- power_mean: calculates raw and normalised subcommunity alpha, rho or gamma diversity by taking the powermean of diversity components
- relativeentropy: calculates raw or normalised subcommunity beta diversity by taking the relative entropy of diversity components
- metacommunity: calculates all subcommunity measures of diversity

Value

inddiv() returns a standard output of class rdiv

References

Reeve, R., T. Leinster, C. Cobbold, J. Thompson, N. Brummitt, S. Mitchell, and L. Matthews. 2016. How to partition diversity. arXiv 1404.6520v3:1–9.

See Also

subdiv for subcommunity-level diversity and metadiv for metacommunity-level diversity.

Examples

```
# Define metacommunity
pop <- cbind.data.frame(A = c(1,1), B = c(2,0), C = c(3,1))
row.names(pop) <- paste0("sp", 1:2)
pop <- pop/sum(pop)
meta <- metacommunity(pop)

# Calculate subcommunity gamma diversity (takes the power mean)
g <- raw_gamma(meta)
inddiv(g, 0:2)

# Calculate subcommunity beta diversity (takes the relative entropy)
b <- raw_beta(meta)
inddiv(b, 0:2)

# Calculate all measures of individual diversity
inddiv(meta, 0:2)
```

`metacommunity` *Metacommunity*

Description

Functions to generate a `metacommunity` object.

Usage

```
metacommunity(partition, similarity)

## S4 method for signature 'data.frame,missing'
metacommunity(partition)

## S4 method for signature 'numeric,missing'
metacommunity(partition)

## S4 method for signature 'matrix,missing'
metacommunity(partition)

## S4 method for signature 'data.frame,matrix'
metacommunity(partition, similarity)

## S4 method for signature 'numeric,matrix'
metacommunity(partition, similarity)

## S4 method for signature 'matrix,matrix'
metacommunity(partition, similarity)
```

```
## S4 method for signature 'missing,similarity'
metacommunity(partition, similarity)

## S4 method for signature 'numeric,similarity'
metacommunity(partition, similarity)

## S4 method for signature 'data.frame,similarity'
metacommunity(partition, similarity)

## S4 method for signature 'matrix,similarity'
metacommunity(partition, similarity)

## S4 method for signature 'ANY,phylo'
metacommunity(partition, similarity)
```

Arguments

partition	two-dimensional matrix of mode numeric with rows as types, columns as subcommunities, and elements containing the relative abundances of types in subcommunities. For phylogenetic diversity, see *Details*
similarity	(optional) object of class similarity

Value

metacommunity() returns an object of class metacommunity (see *Fields*).

Fields

type_abundance two-dimensional matrix of mode numeric with rows as types (species), columns as subcommunities, and each element containing the relative abundance of types in each subcommunity relative to the metacommunity as a whole. In the phylogenetic case, this corresponds to the proportional abundance of historical species, which is calculated from the proportional abundance of terminal taxa

similarity two-dimensional matrix of mode numeric with rows as types, columns as types, and elements containing pairwise similarities between types

similarity_components list containing the components necessary to calculate similarity. This list is empty when precompute_dist = TRUE when calculating distance. When a pairwise distance matrix is too large and precompute_dist = FALSE, this list contains all the information required to calculate pairwise distance between types

similarity_parameters list containing parameters associated with converting pairwise distances to similarities (the dist2sim() arguments)

ordinariness two-dimensional matrix of mode numeric with rows as types, columns as subcommunities, and elements containing the ordinariness of types within subcommunities

subcommunity_weights vector of mode numeric containing subcommunity weights

type_weights two-dimensional matrix of mode numeric, with rows as types, columns as subcommunities, and elements containing weights of types within a subcommunity

`dat_ID` object of class `character` denoting the type of diversity being calculated. This can be "naive", "genetic", "taxonomic", and so on

`raw_abundance` [Phylogenetic] two-dimensional `matrix` of mode `numeric` with rows as types, columns as subcommunities, and elements containing the relative abundance of present day species

`raw_structure` [Phylogenetic] two-dimensional `matrix` of mode `numeric` with rows as historical species, columns as present day species, and elements containing historical species lengths within lineages

`parameters` [Phylogenetic] `data.frame` containing parameters associated with each historic species in the phylogeny

See Also

`metacommunity-class`

Examples

```
# Naive-type
partition <- cbind(a = c(1,1,1,0,0), b = c(0,1,0,1,1))
row.names(partition) <- paste0("sp", 1:5)
partition <- partition / sum(partition)
meta <- metacommunity(partition)
```

---

metacommunity-class *metacommunity-class*

Description

Container for class `metacommunity`.

Usage

```
## S4 method for signature 'metacommunity'
show(object)
```

Arguments

`object`	object of class `metacommunity`

Fields

`type_abundance` two-dimensional `matrix` of mode `numeric` with rows as types (species), columns as subcommunities, and each element containing the relative abundance of types in each subcommunity relative to the metacommunity as a whole. In the phylogenetic case, this corresponds to the proportional abundance of historical species, which is calculated from the proportional abundance of terminal taxa

`similarity` two-dimensional `matrix` of mode `numeric` with rows as types, columns as types, and elements containing the pairwise similarity of types

`similarity_components` list containing the components necessary to calculate similarity. This list is empty when `precompute_dist = TRUE` when calculating distance. When a pairwise distance matrix is too large and `precompute_dist = FALSE`, this list contains all the information required to calculate pairwise distance between types

`similarity_parameters` list containing parameters associated with converting pairwise distances to similarities (the `dist2sim()` arguments)

`ordinariness` two-dimensional `matrix` of mode `numeric` with rows as types, columns as subcommunities, and elements containing the ordinariness of types within subcommunities

`subcommunity_weights` `vector` of mode `numeric` containing subcommunity weights

`type_weights` two-dimensional `matrix` of mode `numeric`, with rows as types, columns as subcommunities, and elements containing weights of types within a subcommunity

`dat_id` object of class `character` describing the class of distance / similarity being used, e.g. "naive", "taxonomic", and so on

`raw_abundance` [Phylogenetic] two-dimensional `matrix` of mode `numeric` with rows as types, columns as subcommunities, and elements containing the relative abundance of present day species

`raw_structure` [Phylogenetic] two-dimensional `matrix` of mode `numeric` with rows as historical species, columns as present day species, and elements containing historical species lengths within lineages

`parameters` [Phylogenetic] `data.frame` containing parameters associated with each historic species in the phylogeny

---

`metadiv` *Metacommunity-level diversity*

Description

Generic function for calculating metacommunity-level diversity.

Usage

```
metadiv(data, qs)

## S4 method for signature 'powermean'
metadiv(data, qs)

## S4 method for signature 'relativeentropy'
metadiv(data, qs)

## S4 method for signature 'metacommunity'
metadiv(data, qs)
```

Arguments

data	matrix of mode numeric; containing diversity components
qs	vector of mode numeric containing q values

Details

data may be input as one of three different classes:

- powermean: raw or normalised metacommunity alpha, rho or gamma diversity components; will calculate metacommunity-level raw or normalised metacommunity alpha, rho or gamma diversity
- relativeentropy: raw or normalised metacommunity beta diversity components; will calculate metacommunity-level raw or normalised metacommunity beta diversity
- metacommunity: will calculate all metacommunity measures of diversity

Value

metadiv() returns a standard output of class rdiv

References

Reeve, R., T. Leinster, C. Cobbold, J. Thompson, N. Brummitt, S. Mitchell, and L. Matthews. 2016. How to partition diversity. arXiv 1404.6520v3:1–9.

See Also

inddiv for type-level diversity and subdiv for subcommunity-level diversity.

Examples

```
# Define metacommunity
pop <- data.frame(a = c(1,3), b = c(1,1))
pop <- pop / sum(pop)
meta <- metacommunity(pop)

# Calculate metacommunity gamma diversity (takes the power mean)
g <- raw_gamma(meta)
metadiv(g, 0:2)

# Calculate metacommunity beta diversity (takes the relative entropy)
b <- raw_beta(meta)
metadiv(b, 0:2)

# Calculate all measures of metacommunity diversity
metadiv(meta, 0:2)
```

meta_gamma *Metacommunity gamma diversity*

Description

Calculates similarity-sensitive metacommunity gamma diversity (the metacommunity similarity-sensitive diversity). This measure may be calculated for a series of orders, represented as a vector of qs.

Usage

```
meta_gamma(meta, qs)
```

Arguments

meta	object of class metacommunity
qs	vector of mode numeric containing q values

Value

meta_gamma returns a standard output of class rdiv

References

R. Reeve, T. Leinster, C. Cobbold, J. Thompson, N. Brummitt, S. Mitchell, and L. Matthews. 2016. How to partition diversity. arXiv 1404.6520v3:1–9.

Examples

```
pop <- data.frame(a = c(1,3), b = c(1,1))
row.names(pop) <- paste0("sp", 1:2)
pop <- pop/sum(pop)
meta <- metacommunity(pop)

# Calculate metacommunity gamma diversity
meta_gamma(meta, 0:2)
```

norm_alpha *Normalised alpha (low level diversity component)*

Description

Calculates the low-level diversity component necessary for calculating normalised alpha diversity.

Usage

```
norm_alpha(meta)
```

Arguments

meta	object of class metacommunity

Details

Values generated from norm_alpha() may be input into subdiv() and metadiv() to calculate normalised subcommunity and metacommunity alpha diversity.

Value

norm_alpha returns an object of class powermean

References

R. Reeve, T. Leinster, C. Cobbold, J. Thompson, N. Brummitt, S. Mitchell, and L. Matthews. 2016. How to partition diversity. arXiv 1404.6520v3:1–9.

Examples

```
pop <- data.frame(a = c(1,3), b = c(1,1))
row.names(pop) <- paste0("sp", 1:2)
pop <- pop/sum(pop)
meta <- metacommunity(pop)

# Calculate normalised alpha component
a <- norm_alpha(meta)
subdiv(a, 1)
metadiv(a, 1)
```

norm_beta *Normalised beta (low level diversity component)*

Description

Calculates the low-level diversity component necessary for calculating normalised beta diversity.

Usage

```
norm_beta(meta)
```

Arguments

meta	object of class metacommunity

Details

Values generated from norm_beta() may be input into subdiv() and metadiv() to calculate normalised subcommunity and metacommunity beta diversity.

Value

norm_beta returns an object of class relativeentropy

References

R. Reeve, T. Leinster, C. Cobbold, J. Thompson, N. Brummitt, S. Mitchell, and L. Matthews. 2016. How to partition diversity. arXiv 1404.6520v3:1–9.

Examples

```
pop <- data.frame(a = c(1,3), b = c(1,1))
row.names(pop) <- paste0("sp", 1:2)
pop <- pop/sum(pop)
meta <- metacommunity(pop)

# Calculate normalised beta component
b <- norm_beta(meta)
subdiv(b, 1)
metadiv(b, 1)
```

`norm_meta_alpha` *Normalised metacommunity alpha diversity*

Description

Calculates similarity-sensitive normalised metacommunity alpha diversity (the average similarity-sensitive diversity of subcommunities). This measure may be calculated for a series of orders, represented as a vector of `qs`.

Usage

```
norm_meta_alpha(meta, qs)
```

Arguments

`meta`	object of class `metacommunity`
`qs`	`vector` of mode `numeric` containing q values

Value

`norm_meta_alpha` returns a standard output of class `rdiv`

References

R. Reeve, T. Leinster, C. Cobbold, J. Thompson, N. Brummitt, S. Mitchell, and L. Matthews. 2016. How to partition diversity. arXiv 1404.6520v3:1–9.

Examples

```
pop <- data.frame(a = c(1,3), b = c(1,1))
row.names(pop) <- paste0("sp", 1:2)
pop <- pop/sum(pop)
meta <- metacommunity(pop)

# Calculate normalised metacommunity alpha diversity
norm_meta_alpha(meta, 0:2)
```

norm_meta_beta *Normalised metacommunity beta diversity*

Description

Calculates similarity-sensitive normalised metacommunity beta diversity (the effective number of distinct subcommunities. This measure may be calculated for a series of orders, represented as a vector of qs.

Usage

```
norm_meta_beta(meta, qs)
```

Arguments

meta	object of class metacommunity
qs	vector of mode numeric containing q values

Value

norm_meta_beta returns a standard output of class rdiv

References

R. Reeve, T. Leinster, C. Cobbold, J. Thompson, N. Brummitt, S. Mitchell, and L. Matthews. 2016. How to partition diversity. arXiv 1404.6520v3:1–9.

Examples

```
pop <- data.frame(a = c(1,3), b = c(1,1))
row.names(pop) <- paste0("sp", 1:2)
pop <- pop/sum(pop)
meta <- metacommunity(pop)

# Calculate normalised metacommunity beta diversity
norm_meta_beta(meta, 0:2)
```

`norm_meta_rho` *Normalised metacommunity rho diversity*

Description

Calculates similarity-sensitive normalised metacommunity rho diversity (the average representativeness of subcommunities. This measure may be calculated for a series of orders, represented as a vector of `qs`.

Usage

```
norm_meta_rho(meta, qs)
```

Arguments

`meta`	object of class `metacommunity`
`qs`	`vector` of mode `numeric` containing q values

Value

`norm_meta_rho` returns a standard output of class `rdiv`

References

R. Reeve, T. Leinster, C. Cobbold, J. Thompson, N. Brummitt, S. Mitchell, and L. Matthews. 2016. How to partition diversity. arXiv 1404.6520v3:1–9.

Examples

```
pop <- data.frame(a = c(1,3), b = c(1,1))
row.names(pop) <- paste0("sp", 1:2)
pop <- pop/sum(pop)
meta <- metacommunity(pop)

# Calculate normalised metacommunity rho diversity
norm_meta_rho(meta, 0:2)
```

norm_rho *Normalised rho (low level diversity component)*

Description

Calculates the low-level diversity component necessary for calculating normalised rho diversity.

Usage

```
norm_rho(meta)
```

Arguments

meta	object of class metacommunity

Details

Values generated from norm_rho() may be input into subdiv() and metadiv() to calculate normalised subcommunity and metacommunity rho diversity.

Value

norm_rho returns an object of class powermean

References

R. Reeve, T. Leinster, C. Cobbold, J. Thompson, N. Brummitt, S. Mitchell, and L. Matthews. 2016. How to partition diversity. arXiv 1404.6520v3:1–9.

Examples

```
pop <- data.frame(a = c(1,3), b = c(1,1))
row.names(pop) <- paste0("sp", 1:2)
pop <- pop/sum(pop)
meta <- metacommunity(pop)

# Calculate normalised rho component
r <- norm_rho(meta)
subdiv(r, 1)
metadiv(r, 1)
```

norm_sub_alpha *Normalised subcommunity alpha diversity*

Description

Calculates similarity-sensitive normalised subcommunity alpha diversity (the diversity of subcommunity *j* in isolation. This measure may be calculated for a series of orders, represented as a vector of qs.

Usage

```
norm_sub_alpha(meta, qs)
```

Arguments

meta	object of class metacommunity
qs	vector of mode numeric containing q values

Value

norm_sub_alpha returns a standard output of class rdiv

References

R. Reeve, T. Leinster, C. Cobbold, J. Thompson, N. Brummitt, S. Mitchell, and L. Matthews. 2016. How to partition diversity. arXiv 1404.6520v3:1–9.

Examples

```
pop <- data.frame(a = c(1,3), b = c(1,1))
row.names(pop) <- paste0("sp", 1:2)
pop <- pop/sum(pop)
meta <- metacommunity(pop)

# Calculate normalised subcommunity alpha diversity
norm_sub_alpha(meta, 0:2)
```

norm_sub_beta *Normalised subcommunity beta diversity*

Description

Calculates similarity-sensitive normalised subcommunity beta diversity (an estimate of the effective number of distinct subcommunities). This measure may be calculated for a series of orders, represented as a vector of qs.

Usage

```
norm_sub_beta(meta, qs)
```

Arguments

meta	object of class metacommunity
qs	vector of mode numeric containing q values

Value

norm_sub_beta returns a standard output of class rdiv

References

R. Reeve, T. Leinster, C. Cobbold, J. Thompson, N. Brummitt, S. Mitchell, and L. Matthews. 2016. How to partition diversity. arXiv 1404.6520v3:1–9.

Examples

```
pop <- data.frame(a = c(1,3), b = c(1,1))
row.names(pop) <- paste0("sp", 1:2)
pop <- pop/sum(pop)
meta <- metacommunity(pop)

# Calculate normalised subcommunity beta diversity
norm_sub_beta(meta, 0:2)
```

norm_sub_rho *Normalised subcommunity rho diversity*

Description

Calculates similarity-sensitive normalised subcommunity rho diversity (the representativeness of subcommunity *j*). This measure may be calculated for a series of orders, represented as a vector of qs.

Usage

```
norm_sub_rho(meta, qs)
```

Arguments

meta	object of class metacommunity
qs	vector of mode numeric containing q values

Value

norm_sub_rho returns a standard output of class rdiv

References

R. Reeve, T. Leinster, C. Cobbold, J. Thompson, N. Brummitt, S. Mitchell, and L. Matthews. 2016. How to partition diversity. arXiv 1404.6520v3:1–9.

Examples

```
pop <- data.frame(a = c(1,3), b = c(1,1))
row.names(pop) <- paste0("sp", 1:2)
pop <- pop/sum(pop)
meta <- metacommunity(pop)

# Calculate normalised subcommunity rho diversity
norm_sub_rho(meta, 0:2)
```

phy2branch *Phylogenetic similarity*

Description

Packages all inputs into an object of class `similarity`.

Usage

```
phy2branch(tree, partition, depth = 1)
```

Arguments

`tree`	object of class `phylo`.
`partition`	two-dimensional `matrix` of mode `numeric` with rows as types (terminal taxa), columns as subcommunities, and each element containing the relative abundance of types in each subcommunity relative to the metacommunity as a whole.
`depth`	proportion of total tree height to be conserved (taken as a proportion from the highest tip). Describes how much evolutionary history should be retained, with 0 marking the date of the most recent tip, and 1 (the default) marking the most recent common ancestor. Numbers greater than 1 extend the root of the tree.

Value

`phy2branch()` returns an object of class `similarity`.

phy2dist *Phylogenetic pairwise tip distance matrix*

Description

Converts any `phylo` object to a matrix of pairwise tip-to-tip distances.

Usage

```
phy2dist(tree, precompute_dist = TRUE)
```

Arguments

`tree`	object of class `phylo`.
`precompute_dist`	object of class `logical` or `numeric`. When TRUE (by default) a distance matrix is generated and stored in slot `distance`, when FALSE no distance matrix is generated, and when numeric a distance matrix is generated until the number of species exceeds the defined value.

Value

phy2sim(x) returns an object of class distance containing a matrix of pairwise tip-to-tip distances.

---

phy_abundance *Relative abundance of historical species*

Description

Calculates the relative abundance of historical species.

Usage

```
phy_abundance(partition, structure_matrix)
```

Arguments

partition	two-dimensional matrix of mode numeric with rows as types (species), columns as subcommunities, and each element containing the relative abundance of types in each subcommunity relative to the metacommunity as a whole. In the phylogenetic case, this corresponds to the proportional abundance of historical species, which is calculated from the proportional abundance of terminal taxa
structure_matrix	output$structure of phy_struct().

---

phy_struct *Calculate phylogenetic structure matrix*

Description

Converts an object into class phylo into class phy_struct.

Usage

```
phy_struct(tree, partition)
```

Arguments

tree	object of class phylo
partition	two-dimensional matrix of mode numeric with rows as types (species), columns as subcommunities, and each element containing the relative abundance of types in each subcommunity relative to the metacommunity as a whole. In the phylogenetic case, this corresponds to the proportional abundance of historical species, which is calculated from the proportional abundance of terminal taxa

Value

`phy_struct()` returns a `list` containing:

$structure - each row denotes historical species, columns denote term
$tbar - the average distance from root to tip for all terminal taxa
$parameters - information associated with each historical species
$tree - object of class phylo

---

powermean *Calculate power mean*

Description

Functions to coerce an object into a powermean (raw_alpha(), norm_alpha(), raw_rho(), norm_rho(), and/or raw_gamma()).

Usage

```
powermean(results, meta, tag)

## S4 method for signature 'powermean'
show(object)
```

Arguments

results	data.frame containing rdiversity outputs associated with norm_alpha(), raw_alpha(), raw_rho(), norm_rho(), and/or raw_gamma()
meta	object of class metacommunity containing the proportional abundance of types, pair-wise similarity, and other associated variables
tag	object of class character naming the diversity measure being calculated
object	object of class powermean

Value

powermean(x) returns an object of class powermean.

print(x) prints an object object of class powermean

Fields

results data.frame containing rdiversity outputs associated with norm_alpha(), raw_alpha(), raw_rho(), norm_rho(), and/or raw_gamma()

measure object of class character naming the diversity measure being calculated

type_abundance two-dimensional matrix of mode numeric with rows as types (species), columns as subcommunities, and each element containing the relative abundance of types in each subcommunity relative to the metacommunity as a whole. In the phylogenetic case, this corresponds to the proportional abundance of historical species, which is calculated from the proportional abundance of terminal taxa

`ordinariness` two-dimensional `matrix` of mode `numeric` with rows as types, columns as subcommunities, and elements containing the ordinariness of types within subcommunities

`subcommunity_weights` `vector` of mode `numeric` containing subcommunity weights

`type_weights` two-dimensional `matrix` of mode `numeric`, with rows as types, columns as subcommunities, and elements containing weights of types within a subcommunity

`dat_id` object of class `character` describing the class of distance / similarity being used, e.g. "naive", "taxonomic", and so on

`similarity_components` list containing the components necessary to calculate similarity. This list is empty when `precompute_dist = TRUE` when calculating distance. When a pairwise distance matrix is too large and `precompute_dist = FALSE`, this list contains all the information required to calculate pairwise distance between types

`similarity_parameters` list containing parameters associated with converting pairwise distances to similarities (the `dist2sim()` arguments)

Examples

```
pop <- data.frame(a = c(1,3), b = c(1,1))
row.names(pop) <- paste0("sp", 1:2)
pop <- pop/sum(pop)
meta <- metacommunity(pop)

# Calculate subcommunity raw alpha diversity (takes the powermean)
a <- raw_alpha(meta)
class(a)
```

`powermean-class` *powermean-class*

Description

Container for class `powermean`.

Fields

`results` `data.frame` containing rdiversity output

`measure` object of class `character` naming the diversity measure being calculated

`type_abundance` two-dimensional `matrix` of mode `numeric` with rows as types (species), columns as subcommunities, and each element containing the relative abundance of types in each subcommunity relative to the metacommunity as a whole. In the phylogenetic case, this corresponds to the proportional abundance of historical species, which is calculated from the proportional abundance of terminal taxa

`ordinariness` two-dimensional `matrix` of mode `numeric` with rows as types, columns as subcommunities, and elements containing the ordinariness of types within subcommunities

`subcommunity_weights` `vector` of mode `numeric` containing subcommunity weights

`type_weights` two-dimensional `matrix` of mode `numeric`, with rows as types, columns as subcommunities, and elements containing weights of types within a subcommunity

`dat_id` object of class `character` describing the class of distance / similarity being used, e.g. "naive", "taxonomic", and so on

`similarity_components` list containing the components necessary to calculate similarity. This list is empty when `precompute_dist = TRUE` when calculating distance. When a pairwise distance matrix is too large and `precompute_dist = FALSE`, this list contains all the information required to calculate pairwise distance between types

`similarity_parameters` list containing parameters associated with converting pairwise distances to similarities (the `dist2sim()` arguments)

`power_mean` *Power mean of vector elements*

Description

`power_mean()` calculates the power mean of a set of values.

Usage

```
power_mean(values, order = 1, weights = rep(1, length(values)))
```

Arguments

`values`	Values for which to calculate mean.
`order`	Order of power mean.
`weights`	Weights of elements, normalised to 1 inside function.

Details

Calculates the order-th power mean of a single set of non-negative values, weighted by weights; by default, weights are equal and order is 1, so this is just the arithmetic mean. Equal weights and a order of 0 gives the geometric mean, and an order of -1 gives the harmonic mean.

Value

Weighted power mean

Examples

```
values <- sample(1:50, 5)
power_mean(values)
```

raw_alpha *Raw alpha (low level diversity component)*

Description

Calculates the low-level diversity component necessary for calculating alpha diversity.

Usage

```
raw_alpha(meta)
```

Arguments

meta	object of class metacommunity

Details

Values generated from raw_alpha() may be input into subdiv() and metadiv() to calculate raw subcommunity and metacommunity alpha diversity.

Value

raw_alpha returns an object of class powermean

References

R. Reeve, T. Leinster, C. Cobbold, J. Thompson, N. Brummitt, S. Mitchell, and L. Matthews. 2016. How to partition diversity. arXiv 1404.6520v3:1–9.

Examples

```
pop <- data.frame(a = c(1,3), b = c(1,1))
row.names(pop) <- paste0("sp", 1:2)
pop <- pop/sum(pop)
meta <- metacommunity(pop)

# Calculate raw alpha component
a <- raw_alpha(meta)
subdiv(a, 1)
metadiv(a, 1)
```

raw_beta *Raw beta (low level diversity component)*

Description

Calculates the low-level diversity component necessary for calculating raw beta diversity.

Usage

```
raw_beta(meta)
```

Arguments

meta	object of class metacommunity

Details

Values generated from raw_beta() may be input into subdiv() and metadiv() to calculate raw subcommunity and metacommunity beta diversity.

Value

raw_beta returns an object of class relativeentropy

References

R. Reeve, T. Leinster, C. Cobbold, J. Thompson, N. Brummitt, S. Mitchell, and L. Matthews. 2016. How to partition diversity. arXiv 1404.6520v3:1–9.

Examples

```
pop <- data.frame(a = c(1,3), b = c(1,1))
row.names(pop) <- paste0("sp", 1:2)
pop <- pop/sum(pop)
meta <- metacommunity(pop)

# Calculate raw beta component
b <- raw_beta(meta)
subdiv(b, 1)
metadiv(b, 1)
```

raw_gamma *Gamma (low level diversity component)*

Description

Calculates the low-level diversity component necessary for calculating gamma diversity.

Usage

```
raw_gamma(meta)
```

Arguments

meta	object of class metacommunity

Details

Values generated from raw_gamma() may be input into subdiv() and metadiv() to calculate subcommunity and metacommunity gamma diversity.

Value

raw_gamma returns an object of class powermean

References

R. Reeve, T. Leinster, C. Cobbold, J. Thompson, N. Brummitt, S. Mitchell, and L. Matthews. 2016. How to partition diversity. arXiv 1404.6520v3:1–9.

Examples

```
pop <- cbind.data.frame(A = c(1,1), B = c(2,0), C = c(3,1))
row.names(pop) <- paste0("sp", 1:2)
pop <- pop/sum(pop)
meta <- metacommunity(pop)

# Calculate gamma component
g <- raw_gamma(meta)
subdiv(g, 1)
metadiv(g, 1)
```

raw_meta_alpha *Raw metacommunity alpha diversity*

Description

Calculates similarity-sensitive raw metacommunity alpha diversity (the naive-community metacommunity diversity). This measure may be calculated for a series of orders, represented as a vector of qs.

Usage

```
raw_meta_alpha(meta, qs)
```

Arguments

meta	object of class metacommunity
qs	vector of mode numeric containing q values

Value

raw_meta_alpha returns a standard output of class rdiv

References

R. Reeve, T. Leinster, C. Cobbold, J. Thompson, N. Brummitt, S. Mitchell, and L. Matthews. 2016. How to partition diversity. arXiv 1404.6520v3:1–9.

Examples

```
pop <- data.frame(a = c(1,3), b = c(1,1))
row.names(pop) <- paste0("sp", 1:2)
pop <- pop/sum(pop)
meta <- metacommunity(pop)

# Calculate raw metacommunity alpha diversity
raw_meta_alpha(meta, 0:2)
```

raw_meta_beta *Raw metacommunity beta diversity*

Description

Calculates similarity-sensitive raw metacommunity beta diversity (the average distinctiveness of subcommunities). This measure may be calculated for a series of orders, represented as a vector of qs.

Usage

```
raw_meta_beta(meta, qs)
```

Arguments

meta	object of class metacommunity
qs	vector of mode numeric containing q values

Value

raw_meta_beta returns a standard output of class rdiv

References

R. Reeve, T. Leinster, C. Cobbold, J. Thompson, N. Brummitt, S. Mitchell, and L. Matthews. 2016. How to partition diversity. arXiv 1404.6520v3:1–9.

Examples

```
pop <- data.frame(a = c(1,3), b = c(1,1))
row.names(pop) <- paste0("sp", 1:2)
pop <- pop/sum(pop)
meta <- metacommunity(pop)

# Calculate raw metacommunity beta diversity
raw_meta_beta(meta, 0:2)
```

raw_meta_rho *Raw metacommunity rho diversity*

Description

Calculates similarity-sensitive raw metacommunity rho diversity (the average redundancy of subcommunities. This measure may be calculated for a series of orders, represented as a vector of qs.

Usage

```
raw_meta_rho(meta, qs)
```

Arguments

meta	object of class metacommunity
qs	vector of mode numeric containing q values

Value

raw_meta_rho returns a standard output of class rdiv

References

R. Reeve, T. Leinster, C. Cobbold, J. Thompson, N. Brummitt, S. Mitchell, and L. Matthews. 2016. How to partition diversity. arXiv 1404.6520v3:1–9.

Examples

```
pop <- data.frame(a = c(1,3), b = c(1,1))
row.names(pop) <- paste0("sp", 1:2)
pop <- pop/sum(pop)
meta <- metacommunity(pop)

# Calculate metacommunity rho diversity
raw_meta_rho(meta, 0:2)
```

raw_rho *Raw rho (low level diversity component)*

Description

Calculates the low-level diversity component necessary for calculating raw rho diversity.

Usage

```
raw_rho(meta)
```

Arguments

meta	object of class metacommunity

Details

Values generated from raw_rho() may be input into subdiv() and metadiv() to calculate raw subcommunity and metacommunity rho diversity.

Value

raw_rho returns an object of class powermean

References

R. Reeve, T. Leinster, C. Cobbold, J. Thompson, N. Brummitt, S. Mitchell, and L. Matthews. 2016. How to partition diversity. arXiv 1404.6520v3:1–9.

Examples

```
pop <- data.frame(a = c(1,3), b = c(1,1))
row.names(pop) <- paste0("sp", 1:2)
pop <- pop/sum(pop)
meta <- metacommunity(pop)

# Calculate raw rho component
r <- raw_rho(meta)
subdiv(r, 1)
metadiv(r, 1)
```

raw_sub_alpha *Raw subcommunity alpha diversity*

Description

Calculates similarity sensitive raw subcommunity alpha diversity (an estimate of naive-community metacommunity diversity). This measure may be calculated for a series of orders, represented as a vector of qs.

Usage

```
raw_sub_alpha(meta, qs)
```

Arguments

meta	object of class metacommunity
qs	vector of mode numeric containing q values

Value

raw_sub_alpha returns a standard output of class rdiv

References

R. Reeve, T. Leinster, C. Cobbold, J. Thompson, N. Brummitt, S. Mitchell, and L. Matthews. 2016. How to partition diversity. arXiv 1404.6520v3:1–9.

Examples

```
pop <- data.frame(a = c(1,3), b = c(1,1))
row.names(pop) <- paste0("sp", 1:2)
pop <- pop/sum(pop)
meta <- metacommunity(pop)

# Calculate raw subcommunity alpha diversity
raw_sub_alpha(meta, 0:2)
```

raw_sub_beta *Raw subcommunity beta diversity*

Description

Calculates similarity-sensitive raw subcommunity beta diversity (the distinctiveness of subcommunity *j*). This measure may be calculated for a series of orders, represented as a vector of qs.

Usage

```
raw_sub_beta(meta, qs)
```

Arguments

meta	object of class metacommunity
qs	vector of mode numeric containing q values

Value

raw_sub_beta returns a standard output of class rdiv

References

R. Reeve, T. Leinster, C. Cobbold, J. Thompson, N. Brummitt, S. Mitchell, and L. Matthews. 2016. How to partition diversity. arXiv 1404.6520v3:1–9.

Examples

```
pop <- data.frame(a = c(1,3), b = c(1,1))
row.names(pop) <- paste0("sp", 1:2)
pop <- pop/sum(pop)
meta <- metacommunity(pop)

# Calculate raw subcommunity beta diversity
raw_sub_beta(meta, 0:2)
```

raw_sub_rho *Raw subcommunity rho diversity*

Description

Calculates similarity-sensitive raw subcommunity rho diversity (the redundancy of subcommunity *j*. This measure may be calculated for a series of orders, represented as a vector of qs.

Usage

```
raw_sub_rho(meta, qs)
```

Arguments

meta	object of class metacommunity
qs	vector of mode numeric containing q values

Value

raw_sub_rho returns a standard output of class rdiv

References

R. Reeve, T. Leinster, C. Cobbold, J. Thompson, N. Brummitt, S. Mitchell, and L. Matthews. 2016. How to partition diversity. arXiv 1404.6520v3:1–9.

Examples

```
pop <- data.frame(a = c(1,3), b = c(1,1))
row.names(pop) <- paste0("sp", 1:2)
pop <- pop/sum(pop)
meta <- metacommunity(pop)

# Calculate raw subcommunity rho diversity
raw_sub_rho(meta, 0:2)
```

---

relativeentropy *Calculate relative entropy*

Description

Functions to coerce an object into a relativeentropy (raw_beta() and/or norm_beta()).

Usage

```
relativeentropy(results, meta, tag)

## S4 method for signature 'relativeentropy'
show(object)
```

Arguments

results	data.frame containing rdiversity outputs associated with raw_beta() and/or norm_beta()
meta	object of class metacommunity containing the proportional abundance of types, pair-wise similarity, and other associated variables
tag	object of class character naming the diversity measure being calculated
object	object of class relativeentropy

Value

object of class relativeentropy

Fields

results data.frame containing rdiversity outputs associated with raw_beta() and/or norm_beta()

measure object of class character naming the diversity measure being calculated

type_abundance two-dimensional matrix of mode numeric with rows as types (species), columns as subcommunities, and each element containing the relative abundance of types in each subcommunity relative to the metacommunity as a whole. In the phylogenetic case, this corresponds to the proportional abundance of historical species, which is calculated from the proportional abundance of terminal taxa

ordinariness two-dimensional matrix of mode numeric with rows as types, columns as subcommunities, and elements containing the ordinariness of types within subcommunities

subcommunity_weights vector of mode numeric containing subcommunity weights

type_weights two-dimensional matrix of mode numeric, with rows as types, columns as subcommunities, and elements containing weights of types within a subcommunity

dat_id object of class character describing the class of distance / similarity being used, e.g. "naive", "taxonomic", and so on

similarity_components list containing the components necessary to calculate similarity. This list is empty when precompute_dist = TRUE when calculating distance. When a pairwise distance matrix is too large and precompute_dist = FALSE, this list contains all the information required to calculate pairwise distance between types

similarity_parameters list containing parameters associated with converting pairwise distances to similarities (the dist2sim() arguments)

Examples

```
pop <- data.frame(a = c(1,3), b = c(1,1))
row.names(pop) <- paste0("sp", 1:2)
pop <- pop/sum(pop)
meta <- metacommunity(pop)

# Calculate raw subcommunity beta diversity
a <- raw_beta(meta)
class(a)
```

relativeentropy-class *relativeentropy-class*

Description

Container for class relativeentropy.

Fields

results data.frame containing rdiversity output

measure object of class character naming the diversity measure being calculated

type_abundance two-dimensional matrix of mode numeric with rows as types (species), columns as subcommunities, and each element containing the relative abundance of types in each subcommunity relative to the metacommunity as a whole. In the phylogenetic case, this corresponds to the proportional abundance of historical species, which is calculated from the proportional abundance of terminal taxa

ordinariness two-dimensional matrix of mode numeric with rows as types, columns as subcommunities, and elements containing the ordinariness of types within subcommunities

subcommunity_weights vector of mode numeric containing subcommunity weights

type_weights two-dimensional matrix of mode numeric, with rows as types, columns as subcommunities, and elements containing weights of types within a subcommunity

dat_id object of class character describing the class of distance / similarity being used, e.g. "naive", "taxonomic", and so on

similarity_components list containing the components necessary to calculate similarity. This list is empty when precompute_dist = TRUE when calculating distance. When a pairwise distance matrix is too large and precompute_dist = FALSE, this list contains all the information required to calculate pairwise distance between types

similarity_parameters list containing parameters associated with converting pairwise distances to similarities (the dist2sim() arguments)

repartition *Repartition metacommunity*

Description

Randomly reshuffles the relative abundance of types (*e.g.* species) in a metacommunity (whilst maintaining the relationship between the relative abundance of a particular species across subcommunities). In the case of a phylogenetic metacommunity, the relative abundance of terminal taxa are randomly reshuffled and the relative abundance of types (historical species) are calculated from the resulting partition.

Usage

```
repartition(meta, new_partition)
```

Arguments

meta	object of class metacommunity.
new_partition	two-dimensional matrix of mode numeric with rows as types (species), columns as subcommunities, and each element containing the relative abundance of types in each subcommunity relative to the metacommunity as a whole. In the phylogenetic case, this corresponds to the proportional abundance of terminal taxa. If this argument is missing, all species / tips will be shuffled

Value

repartition() returns an object of class metacommunity

---

similarity *Generate similarity object*

Description

Container for class similarity.

Usage

```
similarity(similarity, dat_id)

## S4 method for signature 'matrix,character'
similarity(similarity, dat_id)

## S4 method for signature 'matrix,missing'
similarity(similarity, dat_id)
```

Arguments

similarity	similarity matrix
dat_id	object of class character denoting the type of diversity being calculated. This can be "naive", "genetic", "taxonomic", and so on

Value

similarity() returns an object of class similarity.

similarity-class *similarity-class*

Description

Container for class `similarity`.

Usage

```
## S4 method for signature 'similarity'
show(object)
```

Arguments

`object`	object of class `similarity`

Fields

`similarity` two-dimensional `matrix` of mode `numeric` with rows as types, columns as types, and elements containing the pairwise similarity of types

`dat_id` object of class `character` describing the class of distance / similarity being used, e.g. "naive", "taxonomic", and so on

`components` list containing the components necessary to calculate similarity. This list is empty when `precompute_dist = TRUE` when calculating distance. When a pairwise distance matrix is too large and `precompute_dist = FALSE`, this list contains all the information required to calculate pairwise distance between types

`parameters` list containing parameters associated with converting pairwise distances to similarities (the `dist2sim()` arguments)

smatrix *Phylogenetic similarity matrix (ultrametric)*

Description

Function to calculate an ultrametric-similarity matrix.

Usage

```
smatrix(ps)
```

Arguments

`ps`	`phy_struct()` output.

Value

Returns an $hSxhS$ matrix; pair-wise ultrametric-similarity of historic species.

subdiv *Calculate subcommunity-level diversity*

Description

Generic function for calculating subcommunity-level diversity.

Usage

```
subdiv(data, qs)

## S4 method for signature 'powermean'
subdiv(data, qs)

## S4 method for signature 'relativeentropy'
subdiv(data, qs)

## S4 method for signature 'metacommunity'
subdiv(data, qs)
```

Arguments

data	matrix of mode numeric; containing diversity components
qs	vector of mode numeric containing q values

Details

data may be input as one of three different classes:

- powermean: raw or normalised metacommunity alpha, rho or gamma diversity components; will calculate subcommunity-level raw or normalised metacommunity alpha, rho or gamma diversity
- relativeentropy: raw or normalised metacommunity beta diversity components; will calculate subcommunity-level raw or normalised metacommunity beta diversity
- metacommunity: will calculate all subcommunity measures of diversity

Value

subdiv() returns a standard output of class rdiv

References

Reeve, R., T. Leinster, C. Cobbold, J. Thompson, N. Brummitt, S. Mitchell, and L. Matthews. 2016. How to partition diversity. arXiv 1404.6520v3:1–9.

See Also

inddiv for type-level diversity and metadiv for metacommunity-level diversity.

Examples

```
# Define metacommunity
pop <- data.frame(a = c(1,3), b = c(1,1))
row.names(pop) <- paste0("sp", 1:2)
pop <- pop/sum(pop)
meta <- metacommunity(pop)

# Calculate subcommunity gamma diversity (takes the power mean)
g <- raw_gamma(meta)
subdiv(g, 0:2)

# Calculate subcommunity beta diversity (takes the relative entropy)
b <- raw_beta(meta)
subdiv(b, 0:2)

# Calculate all measures of subcommunity diversity
subdiv(meta, 0:2)
```

---

sub_gamma *Subcommunity gamma diversity*

Description

Calculates similarity-sensitive subcommunity gamma diversity (the contribution per individual toward metacommunity diversity). This measure may be calculated for a series of orders, represented as a vector of qs.

Usage

```
sub_gamma(meta, qs)
```

Arguments

meta	object of class metacommunity
qs	vector of mode numeric containing q values

Value

sub_gamma returns a standard output of class rdiv

References

R. Reeve, T. Leinster, C. Cobbold, J. Thompson, N. Brummitt, S. Mitchell, and L. Matthews. 2016. How to partition diversity. arXiv 1404.6520v3:1–9.

Examples

```
pop <- data.frame(a = c(1,3), b = c(1,1))
row.names(pop) <- paste0("sp", 1:2)
pop <- pop/sum(pop)
meta <- metacommunity(pop)

# Calculate subcommunity gamma diversity
sub_gamma(meta, 0:2)
```

`summarise` *Summary function*

Description

This function converts columns of an array (each representing community counts) into proportions, so that each column sums to 1.

Usage

```
summarise(populations, normalise = TRUE)
```

Arguments

`populations`	An S x N array whose columns are counts of individuals.
`normalise`	Normalise probability distribution to sum to 1 for each column rather than just along each set.

Value

Returns an array whose columns are proportions.

`tax2dist` *Generate taxonomic distance matrix*

Description

Calculates taxonomic distances between species.

Usage

```
tax2dist(lookup, tax_distance, precompute_dist = TRUE)
```

Arguments

lookup	data.frame with colnames corresponding to nested taxonomic levels, e.g. c('Species', 'Genus', 'Family', 'Subclass')
tax_distance	vector with the distances attributed to taxonomic levels defined in lookup. The highest distance is the distance attributed to species that are not the same at any recorded taxonomic level. e.g. c(Species = 0, Genus = 1, Family = 2, Subclass = 3, Other = 4) from Shimatani.
precompute_dist	object of class logical or numeric. When TRUE (by default) a distance matrix is generated and stored in slot distance, when FALSE no distance matrix is generated, and when numeric a distance matrix is generated until the number of species exceeds the defined value.

Value

tax2dist() returns an object of class distance containing a matrix of pairwise taxonomic distances

References

Shimatani, K. 2001. On the measurement of species diversity incorporating species differences. Oikos 93:135–147.

Examples

```
# Create Lookup table
Species <- c("tenuifolium", "asterolepis", "simplex var.grandiflora", "simplex var.ochnacea")
Genus <- c("Protium", "Quararibea", "Swartzia", "Swartzia")
Family <- c("Burseraceae", "Bombacaceae", "Fabaceae", "Fabaceae")
Subclass <- c("Sapindales", "Malvales", "Fabales", "Fabales")
lookup <- cbind.data.frame(Species, Genus, Family, Subclass)

# Assign values for each level (Shimatani's taxonomic distance)
tax_distance <- c(Species = 0, Genus = 1, Family = 2, Subclass = 3, Other = 4)

# Generate pairwise distances
distance <- tax2dist(lookup, tax_distance)
similarity <- dist2sim(distance, "linear")
```

---

taxfac *taxfac*

Description

taxfac

Usage

```
taxfac(lookup)
```

Arguments

lookup	data.frame with colnames corresponding to nested hierarchical levels, e.g. c('Species', 'Genus', 'Family', 'Subclass')

Examples

```
## Not run:
# Create Lookup table
Species <- c("tenuifolium", "asterolepis", "simplex var.grandiflora", "simplex var.ochnacea")
Genus <- c("Protium", "Quararibea", "Swartzia", "Swartzia")
Family <- c("Burseraceae", "Bombacaceae", "Fabaceae", "Fabaceae")
Subclass <- c("Sapindales", "Malvales", "Fabales", "Fabales")
lookup <- cbind.data.frame(Species, Genus, Family, Subclass)

taxfac(lookup)

## End(Not run)
```

---

taxid *taxid*

Description

Generate taxanomic codes for each species by converting species, genus, family, and subclass into factors

Usage

```
taxid(tax_fac)
```

Arguments

tax_fac	Output of function tax_fac{}.

Examples

```
## Not run:
# Create Lookup table
Species <- c("tenuifolium", "asterolepis", "simplex var.grandiflora", "simplex var.ochnacea")
Genus <- c("Protium", "Quararibea", "Swartzia", "Swartzia")
Family <- c("Burseraceae", "Bombacaceae", "Fabaceae", "Fabaceae")
Subclass <- c("Sapindales", "Malvales", "Fabales", "Fabales")
lookup <- cbind.data.frame(Species, Genus, Family, Subclass)
```

```
tf <- taxfac(lookup)
taxid(tf)

## End(Not run)
```

---

taxmask *taxmask*

Description

taxmask

Usage

```
taxmask(lookup)
```

Arguments

lookup	Lookup table

Examples

```
## Not run:
# Create Lookup table
Species <- c("tenuifolium", "asterolepis", "simplex var.grandiflora", "simplex var.ochnacea")
Genus <- c("Protium", "Quararibea", "Swartzia", "Swartzia")
Family <- c("Burseraceae", "Bombacaceae", "Fabaceae", "Fabaceae")
Subclass <- c("Sapindales", "Malvales", "Fabales", "Fabales")
lookup <- cbind.data.frame(Species, Genus, Family, Subclass)

taxmask(lookup)

## End(Not run)
```

---

taxvec *taxvec*

Description

Calculate the taxonomic similarity of a single species to all other species. Used by `metacommunity()` to generate a similarity matrix line-by-line when one was not precalculated by `tax2dist()`.

Usage

```
taxvec(similarity, row)
```

Arguments

similarity	An object of class similarity (not containing a similarity matrix).
row	integer denoting which row of the similarity matrix is to be calculated.

Examples

```
## Not run:
# Create Lookup table
Species <- c("tenuifolium", "asterolepis", "simplex var.grandiflora", "simplex var.ochnacea")
Genus <- c("Protium", "Quararibea", "Swartzia", "Swartzia")
Family <- c("Burseraceae", "Bombacaceae", "Fabaceae", "Fabaceae")
Subclass <- c("Sapindales", "Malvales", "Fabales", "Fabales")
lookup <- cbind.data.frame(Species, Genus, Family, Subclass)

# Assign values for each level (Shimatani's taxonomic distance)
tax_distance <- c(Species = 0, Genus = 1, Family = 2, Subclass = 3, Other = 4)

dist <- tax2dist(lookup, tax_distance, precompute_dist = FALSE)
similarity <- dist2sim(dist, "linear")
taxvec(similarity, 1)

## End(Not run)
```

---

tbar *Calculate T_bar*

Description

Function to calculate T_bar.

Usage

```
tbar(partition, structure_matrix)
```

Arguments

partition	two-dimensional matrix of mode numeric with rows as types (species), columns as subcommunities, and each element containing the relative abundance of types in each subcommunity relative to the metacommunity as a whole. In the phylogenetic case, this corresponds to the proportional abundance of historical species, which is calculated from the proportional abundance of terminal taxa
structure_matrix	output$structure of phy_struct(); each row denotes historic species, columns denote terminal taxa, and elements contain branch lengths.

`zmatrix` *Similarity matrix*

Description

Function to calculate a phylogenetic similarity matrix.

Usage

```
zmatrix(partition, s, ps)
```

Arguments

`partition`	two-dimensional `matrix` of mode `numeric` with rows as types (species), columns as subcommunities, and each element containing the relative abundance of types in each subcommunity relative to the metacommunity as a whole. In the phylogenetic case, this corresponds to the proportional abundance of terminal taxa
`s`	`smatrix()` output; ultrametric-similarity matrix.
`ps`	`phy_struct()` output.

Value

`zmatrix()` returns an $hSxhS$ matrix; pair-wise similarity of historic species.

Index