Package ‘rdiversity’

October 14, 2022

Type Package
Title Measurement and Partitioning of Similarity-Sensitive Biodiversity
Version 2.2.0
Date 2022-04-29
Maintainer Richard Reeve <richard.reeve@glasgow.ac.uk>
URL https://github.com/boydorr/rdiversity
BugReports https://github.com/boydorr/rdiversity/issues
License GPL-3
Depends R (>= 2.10)
Imports methods, reshape2, stats, stringdist, utils
Suggests ape, testthat, knitr, markdown, rmarkdown, covr
RoxygenNote 7.1.2
Encoding UTF-8
VignetteBuilder knitr
Collate 'ancestral_nodes.R' 'binary.R' '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'
  'descendant_tips.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'
NeedsCompilation: no

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

Repository: CRAN
Date/Publication: 2022-05-06 07:00:02 UTC

R topics documented:

rdiversity-package ................................................. 3
as.binary ............................................................... 4
binAdd ................................................................. 5
binary ................................................................. 6
binSeq ................................................................. 7
byte ................................................................. 7
bytesNeeded .......................................................... 8
chainsaw .............................................................. 8
dist2sim ............................................................... 9
distance ............................................................... 10
distance-class ......................................................... 10
fillUpToBit ............................................................. 11
fillUpToByte .......................................................... 12
gen2dist ............................................................... 12
inndiv ................................................................. 13
is.binary ............................................................. 14
loadAttributes ......................................................... 15
metacommunity ......................................................... 15
metacommunity-class ................................................ 17
metadiv ............................................................... 18
meta_gamma ............................................................ 19
negate ................................................................. 20
norm_alpha ............................................................ 21
norm_beta ............................................................. 22
norm_meta_alpha ....................................................... 23
norm_meta_beta ....................................................... 24
norm_meta_rho ......................................................... 25
norm_rho .............................................................. 26
norm_sub_alpha ......................................................... 27
norm_sub_beta ........................................................ 28
norm_sub_rho .......................................................... 29
Ops.binary ........................................................... 30
phy2branch ............................................................ 30
phy2dist ............................................................ 31
phy_abundance ......................................................... 31
phy_struct ........................................................... 32
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/EcoJulia/Diversity.jl
Description

Converts an integer (Base10) to a binary (Base2) number. It also converts a logical vector to a binary (Base2) number (see examples).

Usage

as.binary(x, signed=FALSE, littleEndian=FALSE, size=2, n=0, logic=FALSE)

Arguments

x: integer or logical vector.
signed: TRUE or FALSE. Unsigned by default. (two’s complement)
littleEndian: if TRUE. Big Endian if FALSE.
size: in Byte. Needed if signed is set. (by default 2 Byte)
n: in Bit. Can be set if unsigned is set to TRUE. (by default 0 Bit = auto)
logic: If set to TRUE, x is expected as logical vector.

Details

The binary number is represented by a logical vector. The bit order usually follows the same endianness as the byte order. No floating-point support. If logic is set to TRUE an integer vector is interpreted as a logical vector (>0 becomes TRUE and 0 becomes FALSE)

- Little Endian (LSB) —> (MSB)
- Big Endian (MSB) <— (LSB)

Auto switch to signed if num < 0.

Value

a vector of class binary.
binAdd

**See Also**

`is.binary` and `binary`

**Examples**

```r
as.binary(0xAF)
as.binary(42)
as.binary(42, littleEndian=TRUE)
as.binary(c(0xAF, 0xBF, 0xFF))
as.binary(c(2,4,8,16,32), signed=TRUE, size=1)
as.binary(-1, signed=TRUE, size=1)
as.binary(1:7, n=3)
as.binary(sample(2^8,3), n=8)
as.binary(c(1,1,0), signed=TRUE, logic=TRUE)
as.binary(c(TRUE,TRUE,FALSE), logic=TRUE)
```

---

**Description**

Adds two binary numbers. \((x + y)\)

**Usage**

`binAdd(x, y)`

**Arguments**

- **x**  
  summand 1 (binary vector)
- **y**  
  summand 2 (binary vector)

**Details**

Little-Endian and unsigned is not supported at the moment. No floating point supported. If x or y is signed the return value will also be signed.

**Value**

The sum of x and y. Returns a binary vector.

**See Also**

`base::as.logical`, `base::is.logical`, `base::raw`

**Examples**

```r
five <- as.binary(5); ten <- as.binary(10);
as.numeric(rdiversity:::binAdd(ten, five))
rdiversity:::binAdd(as.binary(c(0,1), logic=TRUE), as.binary(c(1,0), logic=TRUE))
```
**Description**

Create objects of type binary.

**Usage**

```r
code
```

**Arguments**

- `n` length of vector. Number of bits
- `signed` TRUE or FALSE. Unsigned by default. (two's complement)
- `littleEndian` if TRUE. Big Endian if FALSE.

**Details**

The binary number is represented by a `logical` vector. The bit order usually follows the same endianess as the byte order. How to read:

- Little Endian (LSB) —> (MSB)
- Big Endian (MSB) <— (LSB)

The Big Endian endianess stores its MSB at the lowest adress. The Little Endian endianess stores its MSB at the highest adress.

E.g. `b <- binary(8)`.

- "Big Endian" : LSB at `b[1]` and MSB at `b[8]`.

No floating-point support.

**Value**

a vector of class binary of length n. By default filled with zeros(0).

**See Also**

`as.binary` and `is.binary`.

**Examples**

```r
code
```

---

### binary

*Binary digit.*

**Description**

Create objects of type binary.

**Usage**

```r
code
```

**Arguments**

- `n` length of vector. Number of bits
- `signed` TRUE or FALSE. Unsigned by default. (two's complement)
- `littleEndian` if TRUE. Big Endian if FALSE.

**Details**

The binary number is represented by a `logical` vector. The bit order usually follows the same endianess as the byte order. How to read:

- Little Endian (LSB) —> (MSB)
- Big Endian (MSB) <— (LSB)

The Big Endian endianess stores its MSB at the lowest adress. The Little Endian endianess stores its MSB at the highest adress.

E.g. `b <- binary(8)`.

- "Big Endian" : LSB at `b[1]` and MSB at `b[8]`.

No floating-point support.

**Value**

a vector of class binary of length n. By default filled with zeros(0).

**See Also**

`as.binary` and `is.binary`.

**Examples**

```r
code
```
**binSeq**

**Description**

Binary sequence.

**Usage**

`binSeq(x, ...)`

**Arguments**

- `x` a sequence.
- `...` used for dec2bin().

**Value**

A sequence list of binary digits.

**See Also**

`binary`

**Examples**

```
rdiversity:::binSeq(0:4)
```

---

**byte**

_A simple helper function to return the size of one byte_

**Description**

Used to increase readability

**Usage**

`byte()`

**Value**

The size of one byte (8)

**See Also**

`fillUpToByte`
bytesNeeded

Description

A simple helper function that returns the minimum number of byte needed to hold the amount of n bit.

Usage

bytesNeeded(n)

Arguments

n The number of bit.

Value

The number of minimum byte needed to hold n bit.

See Also

fillUpToByte or byte

Examples

ten <- as.binary(10)
rdiversity:::bytesNeeded(length(ten))

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`.

---

### dist2sim

**Description**

Converts distance objects into similarity objects.

**Usage**

```r
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 \( 1 - k \times \text{dist} \) for non-negative values, or \( \exp(-k \times \text{dist}) \), respectively. If `normalise` is true, then \( \text{dist} = \text{dist}/\text{max}_d \).

**Value**

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

---

**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**

\[
distance() \text{ returns an object of class distance.}
\]

---

**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

fillUpToBit

Description

Fills up the binary number with zeros(0) or ones(1), to the size n in bit.

Usage

fillUpToBit(x, n, value=FALSE)

Arguments

x The binary number to fill up with zeros. (Any binary vector).

n size in bit.

value to fill up with FALSE(0) or fill up with TRUE(1).

Details

No floating point supported.

Value

binary number. A binary vector with the desired size.

See Also

fillUpToByte.

Examples

rdiversity::fillUpToBit(as.binary(c(1,1), logic=TRUE), n=4)
rdiversity::fillUpToBit(as.binary(c(1,0,1), logic=TRUE), n=4, value=FALSE)
**fillUpToByte**

*Fill up Byte (00000000..)*

**Description**

Fills up the binary number with zeros(0) or ones(1), to the size in Byte.

**Usage**

`fillUpToByte(x, size=0, value=FALSE)`

**Arguments**

- `x`: The binary number to fill up with zeros. (Any binary vector).
- `size`: in Byte. 0 = auto (smallest possible Byte).
- `value`: to fill up with FALSE(0) or fill up with TRUE(1).

**Details**

No floating point supported.

**Value**

Binary number. A binary vector with the desired size.

**See Also**

`fillUpToBit`.

**Examples**

```r
diversity::fillUpToByte(as.binary(c(1,1), logic=TRUE), size=2)
diversity::fillUpToByte(as.binary(c(1,0,1), logic=TRUE), size=2, value=FALSE)
```

---

**gen2dist**

*Genetic distance matrix*

**Description**

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

**Usage**

`gen2dist(vcf, biallelic = FALSE)`
Arguments

vcf object of class data.frame.
biallelic logical describing whether the data is biallelic or not (default).

Value

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

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

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

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

# 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)

is.binary 

is Binary Vector

Description
test for object "binary".

Usage
is.binary(x)

Arguments
x object to test.

Value
TRUE or FALSE.

See Also
as.binary and binary
loadAttributes

Description
Helper function load Attributes

Usage
loadAttributes(x, l)

Arguments
x
l

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 '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)
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

```r
# 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)
```

Description

Container for class metacommunity.

Usage

```r
## 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
metadiv

**Description**

Generic function for calculating metacommunity-level diversity.

**Usage**

```r
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


See Also

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

Examples

```r
# 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)
```

---

```r
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

```r
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


Examples

```r
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)
```

Description

Negates the binary number x. Negation x -> -x or -x -> x

Usage

`negate(x)`

Arguments

x

The number to be negated. A binary vector is expected.

Details

An »unsigned« number will be returned as »signed« regardless of whether the value is negative. No floating point supported.

Value

The negated number of x. Returns a binary vector with signed=TRUE

See Also

`switchEndianess` or `fillUpToByte`.
norm_alpha

Examples

```r
everything <- summary(rdiversity:::negate(as.binary(5, signed=TRUE)))
everything <- summary(rdiversity:::negate(as.binary(-5, signed=TRUE)))
everything <- summary(rdiversity:::negate(as.binary(5, signed=FALSE)))
```

### Description

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

### Usage

```r
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

norm_beta

Normalised beta (low level diversity component)

Description

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

Usage

```r
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


Examples

```r
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)
```
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

\texttt{norm\_meta\_alpha(meta, qs)}

Arguments

- \texttt{meta} : object of class metacommunity
- \texttt{qs} : vector of mode numeric containing \textit{q} values

Value

\texttt{norm\_meta\_alpha} returns a standard output of class \texttt{rdiv}

References


Examples

```r
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**

```r
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**


**Examples**

```r
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)
```
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

\texttt{norm\_meta\_rho(meta, qs)}

Arguments

\begin{description}
\item[meta] object of class \texttt{metacommunity}
\item[qs] vector of mode \texttt{numeric} containing \texttt{q} values
\end{description}

Value

\texttt{norm\_meta\_rho} returns a standard output of class \texttt{rdiv}

References


Examples

\begin{verbatim}
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)
\end{verbatim}
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


Examples
```r
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 $q$s.

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


Examples

```r
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)
```
**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 $q$s.

**Usage**

```r
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**


**Examples**

```r
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**

```r
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**


**Examples**

```r
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)
```
**Ops.binary**

*Group Generic Ops*

**Description**

Group generic Ops operators

**Usage**

```r
## S3 method for class 'binary'
Ops(e1, e2)
```

**Arguments**

- `e1`  
- `e2`

---

**phy2branch**

*Phylogenetic similarity*

**Description**

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

**Usage**

```r
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**

```r
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**

```r
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:

- `$structure`: each row denotes historical species, columns denote terminal taxa
- `$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-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.
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 an 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)
```
print.binary  

Print method for binary number.

Description

This method prints the binary number.

Usage

```r
## S3 method for class 'binary'
print(x, ...)
```

Arguments

- `x`  any binary number.
- `...` further arguments.

Value

Output in ones and zeros (binary vector).

See Also

- `summary.binary` provides some additional information.

raw_alpha  

Raw alpha (low level diversity component)

Description

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

Usage

```r
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


Examples

```r
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)
```

Description

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

Usage

```r
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

**Examples**

```r
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**

```r
raw_gamma(meta)
```

**Arguments**

- `meta` object of class `metacommunity`

**Details**

Values generated from `raw_gamma()` may be input into `subdiv()` and `metadiv()` to calculate sub-community and metacommunity gamma diversity.

**Value**

`raw_gamma` returns an object of class `powermean`

**References**


**Examples**

```r
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
```
raw_meta_alpha

\[ g \leftarrow \text{raw\_gamma}(\text{meta}) \]
\[ \text{subdiv}(g, 1) \]
\[ \text{metadiv}(g, 1) \]

---

**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 \( q_s \).

**Usage**

\[ \text{raw\_meta\_alpha}(\text{meta}, q_s) \]

**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**


**Examples**

```r
g <- raw_gamma(meta)
subdiv(g, 1)
metadiv(g, 1)

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**


**Examples**

```r
codes <- data.frame(a = c(1,3), b = c(1,1))
row.names(codes) <- c("sp", "s")
codes$pop <- (codes$pop + 1)/sum(codes$pop)
codes$meta <- metacommunity(codes$pop)

# Calculate raw metacommunity beta diversity
raw_meta_beta(codes$meta, c(0,2))
```
Description

Calculates similarity-sensitive raw metacommunity rho diversity (the average redundancy of sub-communities. 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


Examples

```r
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 (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


Examples

```r
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

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


Examples

```r
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

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 \( q_s \).

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

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

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 \( q_s \).
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


Examples

code snippet

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)

```r
repartition

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
saveAttributes

**Description**

Helper function save Attributes

**Usage**

`saveAttributes(x)`

**Arguments**

- `x` x

**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.
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)

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)
subdiv

## 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


See Also

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

Examples

```r
# 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)
```
**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**

```r
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**


**Examples**

```r
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)
```
### summary.binary

**Summary method for binary number.**

#### Description

This method provides information about the attributes of the binary number.

#### Usage

```r
## S3 method for class 'binary'
summary(object, ...)
```

#### Arguments

- `object` : binary number.
- `...` : further arguments.

#### Value

Contains the following information:

- Signedness : unsigned or signed
- Endianess : Big-Endian or Little-Endian
- `value<0` : negative or positive number
- `Size[bit]` : Size in bit
- `Base10` : Decimal(Base10) number.

#### See Also

- `print.binary`

### switchEndianess

**Switch Endianess.**

#### Description

Switch little-endian to big-endian and vice versa.

#### Usage

```r
switchEndianess(x, stickyBits=FALSE)
```

#### Arguments

- `x` : binary number. Any binary number.
- `stickyBits` : Bits won't change if set TRUE. Only the attribute will be switched.
Value

switch little-endian to big-endian and vice versa.

See Also

fillUpToByte.

Examples

x <- as.binary(c(1,1,0,0), logic=TRUE); print(x); summary(x);
y <- rdiversity:::switchEndianess(x); print(y); summary(y);
y <- rdiversity:::switchEndianess(x, stickyBits=TRUE); print(y); summary(y);

---

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

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)

declaration of distance variable

declaration of similarity variable

distance <- tax2dist(lookup, tax_distance)
similarity <- dist2sim(distance, "linear")
Index

as.binary, 4, 6, 14
binAdd, 5
binary, 5, 6, 7, 14
binSeq, 7
byte, 7, 8
bytesNeeded, 8
chainsaw, 8
dist2sim, 9
distance, 10
distance, matrix, character-method (distance), 10
distance, matrix, missing-method (distance), 10
distance-class, 10
fillUpToBit, 11, 12
fillUpToByte, 7, 8, 11, 12, 20, 50
gen2dist, 12
inddiv, 13, 19, 47
inddiv, metacommunity-method (inddiv), 13
inddiv, powermean-method (inddiv), 13
inddiv, relativeentropy-method (inddiv), 13
is.binary, 5, 6, 14
loadAttributes, 15
meta_gamma, 19
metacommmunity, 15
metacommunity, matrix, missing-method (metacommunity), 15
metacommunity, matrix, similarity-method (metacommunity), 15
metacommunity, matrix-method (metacommunity), 15
metacommunity, missing, similarity-method (metacommunity), 15
metacommunity, numeric, missing-method (metacommunity), 15
metacommunity, numeric, similarity-method (metacommunity), 15
metacommunity, numeric-method (metacommunity), 15
metacommunity, similarity-method (metacommunity), 15
metacommunity-class, 17
metadiv, 14, 18, 47
metadiv, metacommunity-method (metadiv), 18
metadiv, powermean-method (metadiv), 18
metadiv, relativeentropy-method (metadiv), 18
negate, 20
norm_alpha, 21
norm_beta, 22
norm_meta_alpha, 23
norm_meta_beta, 24
norm_meta_rho, 25
norm_rho, 26
norm_sub_alpha, 27
norm_sub_beta, 28
norm_sub_rho, 29
Ops.binary, 30
phy2branch, 30
phy2dist, 31
phy_abundance, 31
phy_struct, 32
power_mean, 33
powermean-class, 32
print.binary, 34, 49

raw_alpha, 34
raw_beta, 35
raw_gamma, 36
raw_meta_alpha, 37
raw_meta_beta, 38
raw_meta_rho, 39
raw_rho, 40
raw_sub_alpha, 41
raw_sub_beta, 42
raw_sub_rho, 42
rdiversity (rdiversity-package), 3
rdiversity-package, 3
relativeentropy-class, 43
repartition, 44

saveAttributes, 45
show,distance-method (distance-class), 10
show,metacommunity-method
  (metacommunity-class), 17
show,similarity-method
  (similarity-class), 46
similarity, 45
similarity,matrix,character-method
  (similarity), 45
similarity,matrix,missing-method
  (similarity), 45
similarity-class, 46
sub_gamma, 48
subdiv, 14, 19, 46
subdiv,metacommunity-method(subdiv), 46
subdiv,powermean-method(subdiv), 46
subdiv,relativeentropy-method(subdiv), 46
summary.binary, 34, 49
switchEndianess, 20, 49

tax2dist, 50