Package 'funrar'

July 22, 2025

Title Functional Rarity Indices Computation

Version 1.5.0

Description Computes functional rarity indices as proposed by Violle et al. (2017) <doi:10.1016/j.tree.2017.02.002>. Various indices can be computed using both regional and local information. Functional Rarity combines both the functional aspect of rarity as well as the extent aspect of rarity. 'funrar' is presented in Grenié et al. (2017) <doi:10.1111/ddi.12629>.

Depends R (>= 3.2.2)

License GPL (>= 2)

Imports cluster, Matrix, methods, stats

URL https://rekyt.github.io/funrar/, https://github.com/Rekyt/funrar

BugReports https://github.com/Rekyt/funrar/issues

RoxygenNote 7.2.1

Suggests ade4, ggplot2, knitr, rmarkdown, testthat (>= 2.99.0), tidytext

VignetteBuilder knitr

Encoding UTF-8

Config/testthat/edition 3

NeedsCompilation no

Author Matthias Grenié [aut, cre] (ORCID: <https://orcid.org/0000-0002-4659-7522>), Pierre Denelle [aut] (ORCID: <https://orcid.org/0000-0001-5037-2281>), Caroline Tucker [aut] (ORCID: <https://orcid.org/0000-0002-4871-2010>), François Munoz [ths] (ORCID: <https://orcid.org/0000-0001-8776-4705>), Cyrille Violle [ths] (ORCID: <https://orcid.org/0000-0002-2471-9226>)

Maintainer Matthias Grenié <matthias.grenie@gmail.com>

Repository CRAN

Date/Publication 2022-09-23 15:50:02 UTC

Contents

`combination_trait_dist` *Compute Multiple distance matrices from a single trait table*

Description

Internal function to compute combinations of distance matrices from a data.frame of traits, using `compute_dist_matrix()`.

Usage

```
combination_trait_dist(traits_table, ...)
```

Arguments

`traits_table`	a data.frame of traits with species in row and traits in columns, **row names** should be **species names**,
`...`	additional arguments supplied to `compute_dist_matrix()`

Value

A list of functional distance matrices, one for each provided trait plus an additional matrix for all traits taken altogether

---

`compute_dist_matrix` *Compute a Functional Dissimilarity Matrix*

Description

Wrapper for `cluster::daisy()` function in `cluster` package, to compute distance matrix of trait between each pair of species present in given `traits_table`, each row represents a species and each column a trait. To be able to compute other metrics `traits_table` must have species name as row names.

Usage

```
compute_dist_matrix(
  traits_table,
  metric = "gower",
  center = FALSE,
  scale = FALSE
)
```

Arguments

`traits_table`	a data.frame of traits with species in row and traits in columns, **row names** should be **species names**,
`metric`	character vector in list `'gower'`, `'manhattan'`, `'euclidean'` defining the type of distance to use (see `cluster::daisy()`), see Details section,
`center`	logical that defines if traits should be centered (only in the case of `'euclidean'` distance)
`scale`	logical that defines if traits should be scaled (only in the case of `'euclidean'` distance)

Details

The functional distance matrix can be computed using any type of distance metric. When traits are both quantitative and qualitative Gower's (Gower, 1971; Podani, 1999) distance can be used. Otherwise, any other distance metric (Euclidean, Manhattan, Minkowski) can be used - as long as the rows and the columns are named following the species. When using mixed data consider also Gower's distance extension by Pavoine et al. (2009). **IMPORTANT NOTE**: in order to get functional rarity indices between 0 and 1, the distance metric has to be scaled between 0 and 1.

Value

A functional distance matrix, **column** and **row** names follow **species name** from `traits_table` row names.

References

Gower, J.C. (1971) A general coefficient of similarity and some of its properties. Biometrics, 857–871.

Podani, J. (1999) Extending Gower's general coefficient of similarity to ordinal characters. Taxon, 331–340.

Pavoine, S., Vallet, J., Dufour, A.-B., Gachet, S., & Daniel, H. (2009) On the challenge of treating various types of variables: application for improving the measurement of functional diversity. Oikos, 118, 391–402.

See Also

`cluster::daisy()` which this function wraps, base `stats::dist()` or `ade4::dist.ktab()` for Pavoine et al. (2009) extension of Gower's distance.

Examples

```
set.seed(1)  # For reproducibility
trait = data.frame(
   sp = paste("sp", 1:5),
   trait_1 = runif(5),
   trait_2 = as.factor(c("A", "A", "A", "B", "B")))

rownames(trait) = trait$sp

dist_mat = compute_dist_matrix(trait[, -1])
```

---

distinctiveness *Functional Distinctiveness on site-species matrix*

Description

Computes functional distinctiveness from a site-species matrix (containing presence-absence or relative abundances) of species with provided functional distance matrix. The sites-species matrix should have **sites** in **rows** and **species** in **columns**, similar to **vegan** package defaults.

Usage

```
distinctiveness(pres_matrix, dist_matrix, relative = FALSE)
```

Arguments

`pres_matrix`	a site-species matrix (presence-absence or relative abundances), with sites in rows and species in columns
`dist_matrix`	a species functional distance matrix

`relative` a logical indicating if distinctiveness should be scaled relatively to the community (scaled by max functional distance among the species of the targeted community)

Details

The Functional Distinctiveness of a species is the average functional distance from a species to all the other in the given community. It is computed as such:

$$D_i = \frac{\sum_{j=0, i \neq j}^{N} d_{ij}}{N-1},$$

with D_i the functional distinctiveness of species i, N the total number of species in the community and d_{ij} the functional distance between species i and species j. **IMPORTANT NOTE**: in order to get functional rarity indices between 0 and 1, the distance metric has to be scaled between 0 and 1.

Value

a similar matrix from provided `pres_matrix` with Distinctiveness values in lieu of presences or relative abundances, species absent from communities will have an `NA` value (see `Note` section)

Note

Absent species should be coded by `0` or `NA` in input matrices.

When a species is alone in its community the functional distinctiveness cannot be computed (denominator = 0 in formula), and its value is assigned as `NaN`.

For speed and memory efficiency sparse matrices can be used as input of the function using `as(pres_matrix, "dgCMatrix")` from the `Matrix` package. (see `vignette("sparse_matrices", package = "funrar")`)

Examples

```
data("aravo", package = "ade4")
# Site-species matrix
mat = as.matrix(aravo$spe)

# Compute relative abundances
mat = make_relative(mat)

# Example of trait table
tra = aravo$traits[, c("Height", "SLA", "N_mass")]
# Distance matrix
dist_mat = compute_dist_matrix(tra)

di = distinctiveness(pres_matrix = mat, dist_matrix = dist_mat)
di[1:5, 1:5]

# Compute distinctiveness for all species in the regional pool
# i.e., with all the species in all the communities
# Here considering each species present evenly in the regional pool
reg_pool = matrix(1, ncol = ncol(mat))
colnames(reg_pool) = colnames(mat)
```

```
row.names(reg_pool) = c("Regional_pool")

reg_di = distinctiveness(reg_pool, dist_mat)
```

`distinctiveness_alt` *Truncated Functional Distinctiveness*

Description

Computes functional distinctiveness from a site-species matrix (containing presence-absence or relative abundances) of species with provided functional distance matrix considering only species **within a given range** in the functional space. Basically species are cutoff when their dissimilarity is above the input threshold. The sites-species matrix should have **sites** in **rows** and **species** in **columns**, similar to **vegan** package defaults.

Usage

```
distinctiveness_alt(pres_matrix, dist_matrix, given_range)
```

Arguments

`pres_matrix`	a site-species matrix (presence-absence or relative abundances), with sites in rows and species in columns
`dist_matrix`	a species functional distance matrix
`given_range`	a numeric indicating the dissimilarity range at which the the other species are considered maximally dissimilar

Details

The Functional Distinctiveness of a species is the average functional distance from a species to all the other in the given community. It is computed as such:

$$D_i(T) = \frac{\sum_{j=1\ ,j\neq i}^{S} \left[\frac{d_{ij}}{T} + \theta(d_{ij} - T)\left(1 - \frac{d_{ij}}{T}\right)\right]}{S-1}$$

with D_i the functional distinctiveness of species i, N the total number of species in the community and d_{ij} the functional distance between species i and species j. T is the chosen maximal range considered. The function $\theta(d_ij - T)$ is an indicator function that returns 1 when $d_{ij} \geq T$ and 0 when $d_{ij} < T$. **IMPORTANT NOTE**: in order to get functional rarity indices between 0 and 1, the distance metric has to be scaled between 0 and 1.

Value

a similar matrix from provided `pres_matrix` with Distinctiveness values in lieu of presences or relative abundances, species absent from communities will have an `NA` value (see `Note` section)

Note

Absent species should be coded by `0` or `NA` in input matrices.

When a species is alone in its community the functional distinctiveness cannot be computed (denominator = 0 in formula), and its value is assigned as `NaN`.

For speed and memory efficiency sparse matrices can be used as input of the function using `as(pres_matrix, "dgCMatrix")` from the `Matrix` package. (see `vignette("sparse_matrices", package = "funrar")`)

`distinctiveness_com` *Functional Distinctiveness for a single community*

Description

Given a stacked data.frame and a distance matrix compute the functional distinctiveness for a single community. Functional distinctiveness relates to the functional "originality" of a species in a community. The closer to 1 the more the species is functionally distinct from the rest of the community. See `distinctiveness()` function or the functional rarity indices vignette included in the package (type `vignette("rarity_indices", package = "funrar")`), for more details on the metric. **IMPORTANT NOTE**: in order to get functional rarity indices between 0 and 1, the distance metric has to be scaled between 0 and 1.

Usage

```
distinctiveness_com(
  com_df,
  sp_col,
  abund = NULL,
  dist_matrix,
  relative = FALSE
)
```

Arguments

`com_df`	a stacked (= tidy) data.frame from a single community with each row representing a species in a community
`sp_col`	a character vector, the name of the species column in `com_df`
`abund`	a character vector, the name of the column containing relative abundances values
`dist_matrix`	a functional distance matrix as given by `compute_dist_matrix()`, with species name as row and column names
`relative`	a logical indicating if distinctiveness should be scaled relatively to the community (scaled by max functional distance among the species of the targeted community)

Value

the same data.frame with the additional **Di** column giving functional distinctiveness values for each species

Caution

This function is meant for internal uses mostly, thus it does not include any tests on inputs and may fail unexpectedly. Please use `distinctiveness_stack()` to avoid input errors.

See Also

`scarcity_com()`, `vignette("rarity_indices", package = "funrar")` and `distinctiveness()` Details section for detail on the index

`distinctiveness_dimensions` *Distinctiveness across combinations of traits*

Description

From a trait data.frame and a site-species matrix compute Distinctiveness (average pairwise functional distance) for each species in each community on each provided trait and on all traits taken altogether.

Usage

```
distinctiveness_dimensions(pres_matrix, traits_table, ...)
```

Arguments

`pres_matrix`	a site-species matrix, with species in rows and sites in columns, containing presence-absence, relative abundances or abundances values
`traits_table`	a data.frame of traits with species in row and traits in columns, **row names** should be **species names**,
`...`	additional arguments supplied to `compute_dist_matrix()`

Value

a list of site-species matrix with functional distinctiveness values per species per site, with elements **Di_X** for distinctiveness computed on trait **X** and **Di_all** for distinctiveness computed on all traits.

See Also

`uniqueness_dimensions()`, `distinctiveness()`, `distinctiveness_stack()` and `compute_dist_matrix()` for additional arguments

Examples

```
data("aravo", package = "ade4")

# Site-species matrix
mat = as.matrix(aravo$spe)
rel_mat = make_relative(mat)

# Example of trait table
tra = aravo$traits[, c("Height", "SLA", "N_mass")]

di_dim = distinctiveness_dimensions(rel_mat, tra)
```

---

distinctiveness_global

Global/Regional Functional Distinctiveness from dissimilarity matrix

---

Description

Given a distance (or dissimilarity) matrix or `dist()` objects compute regional/global level distinctiveness as if all species were present in the same community.

Usage

```
distinctiveness_global(dist_obj, di_name = "global_di")
```

Arguments

`dist_obj`	a functional distance matrix as given by `compute_dist_matrix()`, with species name as row and column names **or** a `dist()` object with species names as `labels()`
`di_name`	a character vector giving the name of the distinctiveness column in the final data.frame (**default**: `global_di`)

Value

a data.frame with two columns: by default `species` that contains the species names and `global_di` that contains the distinctiveness values. The first column that contains species names can renamed based on `dist_obj` `dimnames`, while the second column is renamed through the `di_name` argument.‘

See Also

`vignette("rarity_indices", package = "funrar")` and `distinctiveness()` Details section for detail on the index

`distinctiveness_range` *Alternative Truncated Functional Distinctiveness*

Description

Computes functional distinctiveness from a site-species matrix (containing presence-absence or relative abundances) of species with provided functional distance matrix considering only species **within a given range** in the functional space. The sites-species matrix should have **sites** in **rows** and **species** in **columns**, similar to **vegan** package defaults.

Usage

```
distinctiveness_range(pres_matrix, dist_matrix, given_range, relative = FALSE)
```

Arguments

`pres_matrix`	a site-species matrix (presence-absence or relative abundances), with sites in rows and species in columns
`dist_matrix`	a species functional distance matrix
`given_range`	a numeric indicating the dissimilarity range at which the the influence of other species is not considered anymore
`relative`	a logical indicating if distinctiveness should be scaled relatively to the community (scaled by max functional distance among the species of the targeted community)

Details

The Functional Distinctiveness of a species is the average functional distance from a species to all the other in the given community. It is computed as such:

$$D_i(T) = 1 \ if \ T < min(d_{ij}), D_i(T) = \left(\frac{\sum_{j=1\ ,j\neq i\ ,d_{ij}\leq T}^{S} d_{ij} \times Ab_j}{\sum_{j=1\ ,j\neq i\ ,d_{ij}\leq T}^{S} Ab_j} \right) \times \left(1 - \frac{\sum_{j=1\ ,j\neq i\ ,d_{ij}\leq T}^{S} Ab_j}{N} \right) \ if \ T \geq min(d_{ij}),$$

with D_i the functional distinctiveness of species i, N the total number of species in the community and d_{ij} the functional distance between species i and species j. T is the chosen maximal range considered. When presence-absence are used $Ab_j = 1/N$ and the term $\left(1 - \frac{\sum_{j=1\ ,j\neq i\ ,d_{ij}\leq T}^{S} Ab_j}{N} \right)$ is replaced by 1. **IMPORTANT NOTE**: in order to get functional rarity indices between 0 and 1, the distance metric has to be scaled between 0 and 1.

Value

a similar matrix from provided `pres_matrix` with Distinctiveness values in lieu of presences or relative abundances, species absent from communities will have an `NA` value (see `Note` section)

Note

Absent species should be coded by `0` or `NA` in input matrices.

When a species is alone in its community the functional distinctiveness cannot be computed (denominator = 0 in formula), and its value is assigned as `NaN`.

For speed and memory efficiency sparse matrices can be used as input of the function using `as(pres_matrix, "dgCMatrix")` from the `Matrix` package. (see `vignette("sparse_matrices", package = "funrar")`)

Examples

```
data("aravo", package = "ade4")
# Site-species matrix
mat = as.matrix(aravo$spe)

# Compute relative abundances
mat = make_relative(mat)

# Example of trait table
tra = aravo$traits[, c("Height", "SLA", "N_mass")]
# Distance matrix
dist_mat = compute_dist_matrix(tra)

di = distinctiveness_range(pres_matrix = mat, dist_matrix = dist_mat, 0.2)
di[1:5, 1:5]
```

`distinctiveness_stack` *Functional Distinctiveness on a stacked data.frame*

Description

Compute Functional Distinctiveness for several communities, from a stacked (or tidy) data.frame of communities, with one column for species identity, one for community identity and an optional one for relative abundances. Also needs a species functional distances matrix. Functional distinctiveness relates to the functional "originality" of a species in a community. The closer to 1 the more the species is functionally distinct from the rest of the community. See `distinctiveness()` function or the functional rarity indices vignette included in the package (type `vignette("rarity_indices", package = "funrar")`), for more details on the metric. **IMPORTANT NOTE**: in order to get functional rarity indices between 0 and 1, the distance metric has to be scaled between 0 and 1. You can either use `_stack()` or `_tidy()` functions as they are aliases of one another.

Usage

```
distinctiveness_stack(
  com_df,
  sp_col,
  com,
  abund = NULL,
```

```
  dist_matrix,
  relative = FALSE
)

distinctiveness_tidy(
  com_df,
  sp_col,
  com,
  abund = NULL,
  dist_matrix,
  relative = FALSE
)
```

Arguments

com_df	a stacked (= tidy) data.frame from a single community with each row representing a species in a community
sp_col	a character vector, the name of the species column in com_df
com	a character vector, the column name for communities names
abund	a character vector, the name of the column containing relative abundances values
dist_matrix	a functional distance matrix as given by compute_dist_matrix(), with species name as row and column names
relative	a logical indicating if distinctiveness should be scaled relatively to the community (scaled by max functional distance among the species of the targeted community)

Value

the same data.frame with the additional **Di** column giving functional distinctiveness values for each species

See Also

scarcity_stack(), uniqueness_stack(), restrictedness_stack(); distinctiveness() Details section for detail on the index

Examples

```
data("aravo", package = "ade4")

# Example of trait table
tra = aravo$traits[, c("Height", "SLA", "N_mass")]
# Distance matrix
dist_mat = compute_dist_matrix(tra)

# Site-species matrix converted into data.frame
mat = as.matrix(aravo$spe)
mat = make_relative(mat)
dat = matrix_to_stack(mat, "value", "site", "species")
```

```
dat$site = as.character(dat$site)
dat$species = as.character(dat$species)

di_df = distinctiveness_stack(dat, "species", "site", "value", dist_mat)
head(di_df)
```

`funrar` *Compute all Functional Rarity Indices from Matrices*

Description

From a site-species matrix and functional distance matrix compute all indices included in the package: functional uniqueness (regional, functional), functional distinctiveness (local, functional), geographical restrictedness (regional, extent), scarcity (local, abundance). **Note**: scarcity can only be computed if relative abundances are provided in the site-species matrix.

Usage

```
funrar(pres_matrix, dist_matrix, rel_abund = FALSE)
```

Arguments

`pres_matrix`	a site-species matrix (presence-absence or relative abundances), with sites in rows and species in columns
`dist_matrix`	a species functional distance matrix
`rel_abund`	logical (`TRUE` or `FALSE`) indicating if site-species matrix contain relative abundances values or only presence-absence data (default = `FALSE`)

Value

A list of 3 objects (or 4 if `rel_abund = TRUE`):

Ui a vector containing uniqueness values per species,

Di a site-species matrix with functional distinctiveness values per species per site,

Ri a vector containing geographical restrictedness values per species,

and if `rel_abund = TRUE`,

Si a site-species matrix with scarcity values per species per site.

See Also

`uniqueness()`, `distinctiveness()`, `restrictedness()`, `scarcity()`

`funrar_stack` *Compute all Functional Rarity Indices from stacked data.frames*

Description

From a stacked (= tidy) data.frame and functional distance matrix compute all indices included in the package: functional uniqueness (regional, functional), functional distinctiveness (local, functional), geographical restrictedness (regional, extent), scarcity (local, abundance). **Note**: scarcity can only be computed if relative abundances are provided in the data.frame.

Usage

```
funrar_stack(com_df, sp_col, com, abund = NULL, dist_matrix)
```

Arguments

`com_df`	a stacked (= tidy) data.frame from a single community with each row representing a species in a community
`sp_col`	a character vector, the name of the species column in `com_df`
`com`	a character vector, the column name for communities names
`abund`	a character vector, the name of the column containing relative abundances values
`dist_matrix`	a functional distance matrix as given by `compute_dist_matrix()`, with species name as row and column names

Value

A list of 3 objects (or 4 if abund is not `NULL`):

Ui a vector containing uniqueness values per species,

Di a site-species matrix with functional distinctiveness values per species per site,

Ri a vector containing geographical restrictedness values per species,

and if abund is not `NULL`,

Si a site-species matrix with scarcity values per species per site.

See Also

`uniqueness_stack()`, `distinctiveness_stack()`, `restrictedness_stack()`, `scarcity_stack()`

`is_relative` *Tell if matrix or data.frame has relative abundances*

Description

From an abundance/presence-absence matrix or data.frame tells if it contains relative abundances or absolute abundances. Checks if all abundances are between 1 and 0 but **never checks sum of abundances per community**.

Usage

```
is_relative(given_obj, abund = NULL)
```

Arguments

`given_obj`	abundance or presence-absence matrix, with sites in rows and species in columns, or tidy community data frame
`abund`	name of the column of the provided object that contains the abundances

Value

`TRUE` if the input has relative abundances `FALSE` otherwise

See Also

`make_relative()` to transform matrix into a relative abundance matrix.

Examples

```
data("aravo", package = "ade4")

# Site-species matrix
mat = as.matrix(aravo$spe)
head(mat)[, 1:5]  # Has absolute abundances
rel_mat = make_relative(mat)
head(rel_mat)  # Relative abundances

# Forced to use ':::' becasue function is not exported
funrar:::is_relative(mat)      # FALSE
funrar:::is_relative(rel_mat)  # TRUE
```

make_relative *Relative abundance matrix from absolute abundance matrix*

Description

From an abundance matrix (numbers of individuals of a given species at a site) returns a relative abundance matrix (proportion of individuals of a given species at a given site). This function works also with sparse matrices.

Usage

```
make_relative(abund_matrix)
```

Arguments

abund_matrix	abundance matrix, with sites in rows and species in columns.

Value

Similar shaped matrix as the input but with relative abundances instead

Examples

```
data("aravo", package = "ade4")

# Site-species matrix
mat = as.matrix(aravo$spe)
head(mat)[, 1:5]  # Has absolute abundances
rel_mat = make_relative(mat)
head(rel_mat)  # Relative abundances
```

matrix_to_stack *Matrix to stacked (= tidy) data.frame*

Description

From a matrix with values to a stacked (= tidy) data.frame, exclude NA from given data.frame. If supplied object is not a matrix, try to coerce object to matrix first. `matrix_to_tidy()` is an alias of this function.

Usage

```
matrix_to_stack(
  my_mat,
  value_col = "value",
  row_to_col = names(dimnames(my_mat))[1],
  col_to_col = names(dimnames(my_mat))[2]
)
```

Arguments

my_mat	matrix you want to transform in stacked (= tidy) data.frame
value_col	(optional) character vector to use for value column (default: 'value')
row_to_col	(optional) character vector used for name of column in data.frame corresponding to rows in matrix (default: corresponding dimension name)
col_to_col	(optional) character vector used for name of column in data.frame corresponding to columns in matrix (default: corresponding dimension name)

Value

a stacked (= tidy) data.frame with, a column for row names, one for column names and a third one for the values.

See Also

stack_to_matrix() for the reverse operation

Examples

```
data("aravo", package = "ade4")

# Site-species matrix converted into data.frame
mat = as.matrix(aravo$spe)
dat = matrix_to_stack(mat, "value", "site", "species")
str(dat)
```

---

restrictedness *Geographical Restrictedness on site-species matrix*

Description

Computes geographical restrictedness from a site-species matrix. Geographical restrictedness is an index related to the extent of a species in a given dataset, it is close to 1 when the species is present in only a single site of the dataset (restricted) and close to 0 when the species is present at all sites. It estimates the geographical extent of a species in a dataset. See Details section to have details on the formula used for the computation. The sites-species matrix should have **sites** in **rows** and **species** in **columns**, similar to **vegan** package defaults.

Usage

```
restrictedness(pres_matrix, relative = FALSE)
```

Arguments

`pres_matrix`	a site-species matrix, with species in rows and sites in columns, containing presence-absence, relative abundances or abundances values
`relative`	a logical (default = FALSE), indicating if restrictedness should be computed relative to restrictedness from a species occupying a single site

Details

Geographical Restrictedness aims to measure the regional extent of a species in **funrar** it is computed the simplest way possible: a ratio of the number of sites where a species is present over the total number of sites in the dataset. We take this ratio off 1 to have a index between 0 and 1 that represents how restricted a species is:

$$R_i = 1 - \frac{N_i}{N_tot},$$

where R_i is the geographical restrictedness value, N_i the total number of sites where species i occur and N_tot the total number of sites in the dataset. When `relative = TRUE`, restrictedness is computed relatively to the restrictedness of a species present in a single site:

$$R_i = \frac{R_i}{R_one}$$

$$R_i = \frac{1 - \frac{K_i}{K_tot}}{1 - \frac{1}{K_tot}}$$

$$R_i = \frac{K_tot - K_i}{K_tot - 1}$$

Other approaches can be used to measure the geographical extent (convex hulls, occupancy models, etc.) but for the sake of simplicity only the counting method is implemented in **funrar**.

Value

A stacked data.frame containing species' names and their restrictedness value in the **Ri** column, similar to what `uniqueness()` returns.

Examples

```
data("aravo", package = "ade4")
# Site-species matrix
mat = as.matrix(aravo$spe)
ri = restrictedness(mat)
head(ri)
```

`restrictedness_stack` *Geographical Restrictedness for stacked data.frame*

Description

Compute the geographical restrictedness for each species present in the stacked data.frame. Geographical restrictedness is an index related to the extent of a species in a given dataset, it is close to 1 when the species is present in only a single site of the dataset (restricted) and close to 0 when the species is present at all sites. It estimates the geographical extent of a species in a dataset. See `restrictedness()` for details on restrictedness computation. You can either use `_stack()` or `_tidy()` functions as they are aliases of one another.

Usage

```
restrictedness_stack(com_df, sp_col, com, relative = FALSE)

restrictedness_tidy(com_df, sp_col, com, relative = FALSE)
```

Arguments

`com_df`	a stacked (= tidy) data.frame of communities
`sp_col`	a character vector indicating the name of the species column
`com`	a character vector indicating the name of the community column
`relative`	a logical (default = FALSE), indicating if restrictedness should be computed relative to restrictedness from a species occupying a single site

Value

A stacked data.frame containing species' names and their restrictedness value in the **Ri** column, similar to what `uniqueness_stack()` returns.

See Also

`restrictedness()`, `uniqueness_stack()`

Examples

```
data("aravo", package = "ade4")

# Site-species matrix converted into data.frame
mat = as.matrix(aravo$spe)
dat = matrix_to_stack(mat, "value", "site", "species")
dat$site = as.character(dat$site)
dat$species = as.character(dat$species)
ri_df = restrictedness_stack(dat, "species", "site")
head(ri_df)
```

`scarcity` *Scarcity on site-species matrix*

Description

Computes scarcity from a relative abundance matrix of species. Scarcity is close to 1 when a species is rare in a community and close to 0 when it is abundant. It requires a site-species matrix with relative abundances. See `Details` section for the formula. The sites-species matrix should have **sites** in **rows** and **species** in **columns**, similar to **vegan** package defaults.

Usage

```
scarcity(pres_matrix)
```

Arguments

`pres_matrix`	a site-species matrix, with species in rows and sites in columns, containing **relative abundances** values

Details

The scarcity of species is computed as follow:

$$S_i = \exp -N \log 2A_i,$$

with S_i the scarcity of species i, N the total number of species in the community and A_i the relative abundance of species i in the community. Scarcity is thus a measure of the **local** rarity in terms of abundances. If S_i is close to 1 the species has a very low abundances while if it's close to 0, it is quite abundant in the community.

Value

a similar matrix to `pres_matrix` with scarcity values in *lieu* of relative abundances.

See Also

`vignette("rarity_indices", package = "funrar")` for details on the scarcity metric; `distinctiveness()`, `restrictedness()`, `uniqueness()`

Examples

```
data("aravo", package = "ade4")
# Site-species matrix
mat = as.matrix(aravo$spe)
mat = make_relative(mat)

si = scarcity(pres_matrix = mat)
si[1:5, 1:5]
```

scarcity_com *Scarcity for a single community*

Description

Given a stacked data.frame compute species scarcity. Scarcity measures how abundant is a species locally. Scarcity is close to 1 when a species is rare in a community and close to 0 when it is abundant. See `scarcity()` function or the functional rarity indices vignette included in the package (type `vignette("rarity_indices", package = "funrar")`) for details about the index.

Usage

```
scarcity_com(com_df, sp_col, abund)
```

Arguments

com_df	a stacked (= tidy) data.frame from a single community with each row representing a species in a community
sp_col	a character vector, the name of the species column in com_df
abund	a character vector, the name of the column containing relative abundances values

Value

the same data.frame with the additional **Si** column giving scarcity values for each species

Caution

This function is meant for internal uses mostly, thus it does not include any tests on inputs and may fail unexpectedly. Please use `scarcity_stack()` to avoid input errors.

See Also

`scarcity()` and `vignette("rarity_indices", package = "funrar")` for details on the scarcity metric; `distinctiveness_com()` to compute distinctiveness on a single community

Examples

```
data("aravo", package = "ade4")

# Site-species matrix converted into data.frame
mat = as.matrix(aravo$spe)
mat = make_relative(mat)
dat = matrix_to_stack(mat, "value", "site", "species")
dat$site = as.character(dat$site)
dat$species = as.character(dat$species)

si_df = scarcity_com(subset(dat, site == "AR07"), "species", "value")
head(si_df)
```

scarcity_stack *Scarcity on a stacked data.frame*

Description

Compute scarcity values for several communities. Scarcity computation requires relative abundances. Scarcity is close to 1 when a species is rare in a community and close to 0 when it is abundant. See `scarcity()` function or the functional rarity indices vignette included in the package (type `vignette("rarity_indices", package = "funrar")`) for details about the index. You can either use `_stack()` or `_tidy()` functions as they are aliases of one another.

Usage

```
scarcity_stack(com_df, sp_col, com, abund)

scarcity_tidy(com_df, sp_col, com, abund)
```

Arguments

com_df	a stacked (= tidy) data.frame from a single community with each row representing a species in a community
sp_col	a character vector, the name of the species column in com_df
com	a character vector indicating the column name of communities ID in com_df
abund	a character vector, the name of the column containing relative abundances values

Value

The same table as com_df with an added S_i column for Scarcity values.

See Also

`scarcity()` and `vignette("rarity_indices", package = "funrar")` for details on the scarcity metric; `distinctiveness_stack()`, `restrictedness_stack()`, `uniqueness_stack()`

Examples

```
data("aravo", package = "ade4")

# Site-species matrix converted into data.frame
mat = as.matrix(aravo$spe)
mat = make_relative(mat)
dat = matrix_to_stack(mat, "value", "site", "species")
dat$site = as.character(dat$site)
dat$species = as.character(dat$species)

si_df = scarcity_stack(dat, "species", "site", "value")
head(si_df)
```

stack_to_matrix *Stacked (= tidy) data.frame to matrix*

Description

Passes from a stacked (= tidy) data.frame to a matrix. `tidy_to_matrix()` is an alias of this function.

Usage

```
stack_to_matrix(
  my_df,
  col_to_row,
  col_to_col,
  col_value = NULL,
  sparse = FALSE
)
```

Arguments

`my_df`	data.frame you want to transform in matrix
`col_to_row`	character vector of the name of the data.frame column you want to put into matrix rows
`col_to_col`	character vector of the name of the data.frame column you want to be as columns in matrix
`col_value`	(optional, default = `NULL`) character vector indicating the name of a column coding the values that will be put in the matrix
`sparse`	(optional, default = `FALSE`) logical indicating whether to return a sparse matrix (if `TRUE` requires `tidytext` package)

Value

a matrix with given `col_to_row` column in rows and `col_to_col` column in columns. If some cells are not present in the data.frame (e.g. some species not present at some sites), the matrix will have a `NA` value.

See Also

`matrix_to_stack()` for the reverse operation

Examples

```
example = data.frame("sites" = c(rep("1", 3), rep("2", 2)),
 "species" = c("A", "B", "C", "B", "D"),
  "abundance" = c(0.33, 0.33, 0.33, 0.4, 0.6))

mat = stack_to_matrix(example, "sites", "species", "abundance")
```

mat

---

uniqueness *Functional Uniqueness for site-species matrix matrix*

Description

Computes the functional uniqueness from a site-species matrix with the provided functional distance matrix. Functional Uniqueness represents how "isolated" is a species in the global species pool, it is the functional distance to the nearest neighbor of the species of interest (see `Details` section for the formula). The sites-species matrix should have **sites** in **rows** and **species** in **columns**, similar to **vegan** package defaults.

Usage

```
uniqueness(pres_matrix, dist_matrix)
```

Arguments

`pres_matrix`	a site-species matrix (presence-absence or relative abundances), with sites in rows and species in columns
`dist_matrix`	a species functional distance matrix

Details

Functional Uniqueness U_i is computed as follow:

$$U_i = \min(d_{ij}) \forall j, j \neq i,$$

with U_i the functional uniqueness of species i, and $d_i j$ the functional distance between species i and species j

Value

A data.frame with functional uniqueness values per species, with one column with provided species column name and the **Ui** column with functional uniqueness values.

See Also

`distinctiveness()`, `restrictedness()`, `scarcity()`

Examples

```
data("aravo", package = "ade4")

# Site-species matrix
mat = as.matrix(aravo$spe)
colnames(mat) = as.character(colnames(mat))

# Example of trait table
tra = aravo$traits[, c("Height", "SLA", "N_mass")]
# Distance matrix
dist_mat = compute_dist_matrix(tra)

ui = uniqueness(mat, dist_mat)
head(ui)

# Computing uniqueness for each community
com_ui = apply(mat, 1,
                function(x, dist_m) {
                    smaller_com = x[x > 0 & !is.na(x)]
                    uniqueness(t(as.matrix(smaller_com)), dist_m)
                }, dist_m = dist_mat)
```

`uniqueness_dimensions` *Uniqueness across combinations of traits*

Description

From a trait table and a site-species matrix compute Uniqueness (nearest functional distance) for each species and each trait, plus computes it for all the traits.

Usage

```
uniqueness_dimensions(pres_matrix, traits_table, ...)
```

Arguments

`pres_matrix`	a site-species matrix, with species in rows and sites in columns, containing presence-absence, relative abundances or abundances values
`traits_table`	a data.frame of traits with species in row and traits in columns, **row names** should be **species names**,
`...`	additional arguments supplied to `compute_dist_matrix()`

Value

a data.frame containing species' names and their uniqueness values for each traits (**Ui_X** column for trait **X**), as well as a column for the uniqueness value for all traits (**Ui_all** column)

See Also

distinctiveness_dimensions(), uniqueness(), uniqueness_stack() and compute_dist_matrix() for additional arguments

Examples

```
data("aravo", package = "ade4")

# Site-species matrix
mat = as.matrix(aravo$spe)
rel_mat = make_relative(mat)

# Example of trait table
tra = aravo$traits[, c("Height", "SLA", "N_mass")]

ui_dim = uniqueness_dimensions(rel_mat, tra)
```

---

uniqueness_stack *Functional Uniqueness on stacked data.frame*

Description

Computes functional uniqueness values over a given regional pool. Functional uniqueness gives the functional distance to the nearest-neighbor of a given species in the provided distance matrix. See uniqueness() function for details on computation. You can either use _stack() or _tidy() functions as they are aliases of one another.

Usage

```
uniqueness_stack(com_df, sp_col, dist_matrix)

uniqueness_tidy(com_df, sp_col, dist_matrix)
```

Arguments

com_df	a data frame of the species in the regional pool.
sp_col	a character vector indicating the name of the species column in the com_df data frame
dist_matrix	a functional distance matrix

Value

A data.frame with uniqueness value per species, with one column with provided species column name and the **Ui** column with the uniqueness values.

See Also

uniqueness() and vignette("rarity_indices", package = "funrar") for details on the uniqueness metric; distinctiveness_stack(), restrictedness_stack(), scarcity_stack()

Examples

```
data("aravo", package = "ade4")

# Site-species matrix converted into data.frame
mat = as.matrix(aravo$spe)
dat = matrix_to_stack(mat, "value", "site", "species")
dat$site = as.character(dat$site)
dat$species = as.character(dat$species)

# Example of trait table
tra = aravo$traits[, c("Height", "SLA", "N_mass")]
# Distance matrix
dist_mat = compute_dist_matrix(tra)

ui_df = uniqueness_stack(dat, "species", dist_mat)
head(ui_df)
```

Index