Package 'betapart'

May 7, 2026

Type Package

Title Partitioning Beta Diversity into Turnover and Nestedness Components

Version 1.6.1

Date 2025-07-24

Imports ape, fastmatch, geometry, picante, rcdd, doSNOW, foreach, snow, itertools, minpack.lm

Suggests vegan

Description Functions to compute pair-wise dissimilarities (distance matrices) and multiple-site dissimilarities, separating the turnover and nestedness-resultant components of taxonomic (incidence and abundance based), functional and phylogenetic beta diversity.

License GPL (>= 2)

NeedsCompilation no

Maintainer Andres Baselga <andres.baselga@usc.es>

Author Andres Baselga [aut, cre],
David Orme [aut],
Sebastien Villeger [aut],
Julien De Bortoli [aut],
Fabien Leprieur [aut],
Maxime Logez [aut],
Sara Martinez-Santalla [aut],
Ramiro Martin-Devasa [aut],
Carola Gomez-Rodriguez [aut],
Rosa M. Crujeiras [aut],
Renato Henriques-Silva [ctb]

Repository CRAN

Date/Publication 2025-07-24 15:20:15 UTC

Contents

`bbsData` *BBS data by state for two timeslices*

Description

The data consists of binary presence/absence matrices for 569 bird species across 49 US states for two time slices (1980 - 1985 and 2000 - 2005). Only species (identified by AOU number) recorded during both time periods are included. The data are taken from the North American Breeding Bird Survey dataset and from a version of the database downloaded in May 2009.

Usage

```
data(bbsData)
```

Format

Two matrices (bbs1980 and bbs2000) of identical structure showing the presence/absence of the species as binary data.

`state` US states by USPS two letter codes.

`aou` Species identity by AOU species ID numbers.

Source

http://www.pwrc.usgs.gov/BBS/

Examples

```
data(bbsData)
str(bbs1980)
str(bbs2000)
```

---

beta.multi *Multiple-site dissimilarities*

Description

Computes 3 multiple-site dissimilarities accounting for the spatial turnover and the nestedness components of beta diversity, and the sum of both values

Usage

```
beta.multi(x, index.family="sorensen")
```

Arguments

`x`	data frame, where rows are sites and columns are species. Alternatively `x` can be a `betapart` object derived from the `betapart.core` function
`index.family`	family of dissimilarity indices, partial match of `"sorensen"` or `"jaccard"`.

Value

The function returns a list with the three multiple site dissimilarity values.

For `index.family="sorensen"` the three indices are:

`beta.SIM`	value of the turnover component, measured as Simpson dissimilarity
`beta.SNE`	value of the nestedness component, measured as nestedness-resultant fraction of Sorensen dissimilarity
`beta.SOR`	value of the overall beta diversity, measured as Sorensen dissimilarity

For `index.family="jaccard"` the three indices are:

beta.JTU	value of the turnover component, measured as turnover fraction of Jaccard dissimilarity
beta.JNE	value of the nestedness component, measured as nestedness-resultant fraction of Jaccard dissimilarity
beta.JAC	value of the overall beta diversity, measured as Jaccard dissimilarity

Author(s)

Andrés Baselga and David Orme

References

Baselga, A. 2010. Partitioning the turnover and nestedness components of beta diversity. Global Ecology and Biogeography 19:134-143

Baselga, A. 2012. The relationship between species replacement, dissimilarity derived from nestedness, and nestedness. Global Ecology and Biogeography 21, 1223-1232

See Also

beta.pair, beta.sample, betapart.core, beta.temp

Examples

```
data(ceram.s)
ceram.beta<-beta.multi(ceram.s, index.family="sor")
```

---

beta.multi.abund *Abundance-based multiple-site dissimilarities*

Description

Computes 3 multiple-site dissimilarities accounting for the (i) balanced variation and (ii) abundance gradient components of dissimilarity, and the sum of both values (i.e. total abundance-based dissimilarity)

Usage

```
beta.multi.abund(x, index.family="bray")
```

Arguments

x	data frame, where rows are sites and columns are species. Alternatively x can be a betapart.abund object derived from the betapart.core.abund function
index.family	family of dissimilarity indices, partial match of "bray" or "ruzicka".

Value

The function returns a list with the three multiple site dissimilarity values.

For `index.family="bray"` the three indices are:

`beta.BRAY.BAL`	value of the balanced variation component of Bray-Curtis multiple-site dissimilarity
`beta.BRAY.GRA`	value of the abundance-gradient component of Bray-Curtis multiple-site dissimilarity
`beta.BRAY`	value of the overall dissimilarity, measured as Bray-Curtis multiple-site dissimilarity

For `index.family="ruzicka"` the three indices are:

`beta.RUZ.BAL`	value of the balanced variation component of Ruzicka multiple-site dissimilarity
`beta.RUZ.GRA`	value of the abundance-gradient component of Ruzicka multiple-site dissimilarity
`beta.RUZ`	value of the overall dissimilarity, measured as Ruzicka multiple-site dissimilarity

Author(s)

Andrés Baselga

References

Baselga, A. 2017. Partitioning abundance-based multiple-site dissimilarity into components: balanced variation in abundance and abundance gradients. Methods in Ecology and Evolution 8: 799-808

See Also

`beta.pair.abund`, `beta.sample.abund`, `betapart.core.abund`, `beta.multi`

Examples

```
require(vegan)
data(BCI)
beta.multi.abund(BCI, index.family="bray")
```

`beta.pair` *Incidence-based pair-wise dissimilarities*

Description

Computes 3 distance matrices accounting for the (i) turnover (replacement), (ii) nestedness-resultant component, and (iii) total dissimilarity (i.e. the sum of both components).

Usage

```
beta.pair(x, index.family = "sorensen")
```

Arguments

`x`	data frame, where rows are sites and columns are species. Alternatively `x` can be a `betapart` object derived from the `betapart.core` function
`index.family`	family of dissimilarity indices, partial match of `"sorensen"` or `"jaccard"`.

Value

The function returns a list with three dissimilarity matrices. For `index.family="sorensen"` the three matrices are:

`beta.sim`	`dist` object, dissimilarity matrix accounting for spatial turnover (replacement), measured as Simpson pair-wise dissimilarity
`beta.sne`	`dist` object, dissimilarity matrix accounting for nestedness-resultant dissimilarity, measured as the nestedness-fraction of Sorensen pair-wise dissimilarity
`beta.sor`	`dist` object, dissimilarity matrix accounting for total dissimilarity, measured as Sorensen pair-wise dissimilarity (a monotonic transformation of beta diversity)

For `index.family="jaccard"` the three matrices are:

`beta.jtu`	`dist` dissimilarity matrix accounting for spatial turnover, measured as the turnover-fraction of Jaccard pair-wise dissimilarity
`beta.jne`	`dist` object, dissimilarity matrix accounting for nestedness-resultant dissimilarity, measured as the nestedness-fraction of Jaccard pair-wise dissimilarity
`beta.jac`	`dist` object, dissimilarity matrix accounting for beta diversity, measured as Jaccard pair-wise dissimilarity (a monotonic transformation of beta diversity)

Author(s)

Andrés Baselga and David Orme

References

Baselga, A. 2010. Partitioning the turnover and nestedness components of beta diversity. Global Ecology and Biogeography 19:134-143

Baselga, A. 2012. The relationship between species replacement, dissimilarity derived from nestedness, and nestedness. Global Ecology and Biogeography 21, 1223-1232

See Also

beta.pair.abund, beta.multi, beta.sample, betapart.core, beta.temp

Examples

```
data(ceram.s)
ceram.dist<-beta.pair(ceram.s, index.family="jac")
```

---

beta.pair.abund *Abundance-based pair-wise dissimilarities*

Description

Computes 3 distance matrices accounting for the (i) balanced variation in abundances, (ii) abundance gradients, and (iii) total dissimilarity (i.e. the sum of both components).

Usage

```
beta.pair.abund(x, index.family = "bray")
```

Arguments

x	data frame, where rows are sites and columns are species. Alternatively x can be a betapart.abund object derived from the betapart.core.abund function
index.family	family of dissimilarity indices, partial match of "bray" or "ruzicka".

Value

The function returns a list with three dissimilarity matrices. For index.family="bray" the three matrices are:

beta.bray.bal	dist object, dissimilarity matrix accounting for the dissimilarity derived from balanced variation in abundance between sites
beta.bray.gra	dist object, dissimilarity matrix accounting for the dissimilarity derived from unidirectional abundance gradients
beta.bray	dist object, dissimilarity matrix accounting for total abundance-based dissimilarity between sites, measured as the Bray-Curtis index

For index.family="ruzicka" the three matrices are:

beta.ruz.bal	dist object, dissimilarity matrix accounting for the dissimilarity derived from balanced variation in abundance between sites
beta.ruz.gra	dist object, dissimilarity matrix accounting for the dissimilarity derived from unidirectional abundance gradients
beta.ruz	dist object, dissimilarity matrix accounting for total abundance-based dissimilarity between sites, measured as the Ruzicka index

Author(s)

Andrés Baselga

References

Baselga, A. 2013. Separating the two components of abundance-based dissimilarity: balanced changes in abundance vs. abundance gradients. Methods in Ecology and Evolution 4: 552–557

Legendre, P. 2014. Interpreting the replacement and richness difference components of beta diversity. Global Ecology and Biogeography, 23: 1324–1334

See Also

`beta.multi.abund`, `beta.sample.abund`, `betapart.core.abund`, `beta.pair`

Examples

```
require(vegan)
data(BCI)
BCI.pair<-beta.pair.abund(BCI, index.family="bray")
```

`beta.para.control` *Specifying Control Values for Internal Parallel Cluster*

Description

The values supplied in the `beta.para.control()` call replace the defaults, and a list with all settings (i.e., values for all possible arguments) is returned. The returned list is used to define the internal parallel cluster of the `functional.betapart.core` function.

Usage

```
beta.para.control(nc = floor(parallel::detectCores()/2), type = "SOCK",
                                LB = TRUE, size = 1)
```

Arguments

`nc`	number of cores to use. Default is half of the available cores.
`type`	character - the type of cluster to be used, either `"SOCK"`, `"PSOCK"` or `"FORK"` (not on Windows).
`LB`	logical indicating if load balancing has to be used. Default is `TRUE`
`size`	number of operation run on each core at each iteration. Default is `1`.

Value

a list with components for each of the possible arguments.

Author(s)

Maxime Logez

Examples

```
str(beta.para.control(nc = 2, LB = FALSE))
```

`beta.sample` *Resampling multiple-site dissimilarity for n sites*

Description

Resamples the 3 multiple-site dissimilarities (turnover, nestedness-resultant fraction, and overall beta diversity) for a subset of sites of the original data frame.

Usage

```
beta.sample(x, index.family="sorensen", sites=nrow(x$data), samples = 1)
```

Arguments

`x`	data frame, where rows are sites and columns are species. Alternatively `x` can be a `betapart` object derived from the `betapart.core` function.
`index.family`	family of dissimilarity indices, partial match of `"sorensen"` or `"jaccard"`.
`sites`	number of sites for which multiple-site dissimilarities will be computed. If not specified, default is all sites.
`samples`	number of repetitions. If not specified, default is 1.

Value

The function returns a list with a dataframe with the resampled 3 multiple-site dissimilarities (turnover fraction, nestedness-resultant fraction and overall dissimilarity; see `beta.multi`), a vector with the respective means and a vector with the respective standard deviation.

For `index.family="sorensen"`:

`sampled.values`	dataframe containing beta.SIM, beta.SNE and beta.SOR for all samples
`mean.values`	vector containing the mean values of beta.SIM, beta.SNE and beta.SOR among samples
`sd.values`	vector containing the sd values of beta.SIM, beta.SNE and beta.SOR among samples

For `index.family="jaccard"`:

`sampled.values`	dataframe containing beta.JTU, beta.JNE and beta.JAC for all samples
`mean.values`	vector containing the mean values of beta.JTU, beta.JNE and beta.JAC among samples
`sd.values`	vector containing the sd values of beta.JTU, beta.JNE and beta.JAC among samples

Author(s)

Andrés Baselga and David Orme

References

Baselga, A. 2010. Partitioning the turnover and nestedness components of beta diversity. Global Ecology and Biogeography 19:134-143

Baselga, A. 2012. The relationship between species replacement, dissimilarity derived from nestedness, and nestedness. Global Ecology and Biogeography 21, 1223-1232

See Also

beta.multi, beta.sample, beta.temp

Examples

```
# Read the data for Northern and Southern European cerambycids
data(ceram.s)
data(ceram.n)

# Resample 100 times the multiple-site dissimilarities
# for 10 countries.
beta.ceram.s<-beta.sample(ceram.s, index.family="sor", sites=10, samples=100)
beta.ceram.n<-beta.sample(ceram.n, index.family="sor", sites=10, samples=100)

# Plot the distributions of beta.SIM in Southern Europe (red)
# and Northern Europe (blue)
plot(density(beta.ceram.s$sampled.values$beta.SIM), col="red", xlim=c(0,1))
lines(density(beta.ceram.n$sampled.values$beta.SIM), col="blue")

# Compute the p-value of difference in beta.SIM between South and North
# (i.e. the probability of finding in the North a higher value than
# in the South)
p.value.beta.SIM<-length(which(beta.ceram.s$sampled.values$beta.SIM<
beta.ceram.n$sampled.values$beta.SIM))/100

p.value.beta.SIM
# The result is 0 and we used 100 samples, so p<0.01
```

---

beta.sample.abund *Resampling abundance-based multiple-site dissimilarity for n sites*

Description

Resamples the 3 abundance-based multiple-site dissimilarities (balanced variation fraction,abundance-gradient fraction, and overall dissimilarity) for a subset of sites of the original data frame.

Usage

```
beta.sample.abund(x, index.family="bray", sites = nrow(x), samples = 1)
```

Arguments

x	data frame, where rows are sites and columns are species
index.family	family of dissimilarity indices, partial match of "bray" or "ruzicka".
sites	number of sites for which multiple-site dissimilarities will be computed. If not specified, default is all sites.
samples	number of repetitions. If not specified, default is 1.

Value

The function returns a list with a dataframe with the resampled 3 multiple-site dissimilarities (balanced variation fraction, abundance-gradient fraction and overall dissimilarity; see beta.multi.abund), a vector with the respective means and a vector with the respective standard deviation.

For index.family="bray":

sampled.values	dataframe containing beta.BRAY.BAL, beta.BRAY.GRA and beta.BRAY for all samples
mean.values	vector containing the mean values of beta.BRAY.BAL, beta.BRAY.GRA and beta.BRAY among samples
sd.values	vector containing the sd values of beta.BRAY.BAL, beta.BRAY.GRA and beta.BRAY among samples

For index.family="ruzicka":

sampled.values	dataframe containing beta.RUZ.BAL, beta.RUZ.GRA and beta.RUZ for all samples
mean.values	vector containing the mean values of beta.RUZ.BAL, beta.RUZ.GRA and beta.RUZ among samples
sd.values	vector containing the sd values of beta.RUZ.BAL, beta.RUZ.GRA and beta.RUZ among samples

Author(s)

Andrés Baselga

References

Baselga, A. 2017. Partitioning abundance-based multiple-site dissimilarity into components: balanced variation in abundance and abundance gradients. Methods in Ecology and Evolution 8: 799-808

See Also

beta.multi.abund, beta.sample

Examples

```
require(vegan)
data(BCI)
beta.sample.abund(BCI, index.family="bray", sites=10, samples=100)
```

`beta.temp` *Temporal change in community composition*

Description

Computes the dissimilarity for each locality between time 1 and time 2, considering the turnover and nestedness components of temporal change, and the sum of both values (overall change)

Usage

```
beta.temp(x, y, index.family="sorensen")
```

Arguments

`x`	data frame for time 1, where rows are sites and columns are species
`y`	data frame for time 2, where rows are sites and columns are species. `x` and `y` must contain exactly the same sites and species
`index.family`	family of dissimilarity indices, partial match of `"sorensen"` or `"jaccard"`.

Value

The function returns a data frame where rows are sites and columns are pairwise dissimilarity values between cell composition in time 1 and time 2. For `index.family="sorensen"` the indices are beta.sim, beta.sne, and beta.sor. For `index.family="jaccard"` the indices are beta.jtu, beta.sne, and beta.jac.

Author(s)

Andrés Baselga and David Orme

References

Baselga, A. 2010. Partitioning the turnover and nestedness components of beta diversity. Global Ecology and Biogeography 19:134-143

Baselga, A. 2012. The relationship between species replacement, dissimilarity derived from nestedness, and nestedness. Global Ecology and Biogeography 21, 1223-1232

See Also

`beta.multi`, `beta.pair`, `beta.sample`, `betapart.core`,

Examples

```
data(bbsData)
bbs.t <- beta.temp(bbs1980, bbs2000, index.family="sor")
```

`betapart` *Partitioning beta diversity into turnover and nestedness components*

Description

betapart allows computing pair-wise dissimilarities (distance matrices) and multiple-site dissimilarities, separating the turnover and nestedness-resultant components of taxonomic (incidence and abundance based), functional and phylogenetic beta diversity.

Details

The partitioning of incidence-based dissimilarity can be performed for two different families of indices: Sorensen and Jaccard. The pairwise function `beta.pair` yields 3 distance matrices accounting for the spatial turnover and the nestedness components of beta-diversity. The third distance matrix accounts for the sum of both components, i.e. total dissimilarity (a monotonic transformation of beta diversity). The multiple site function `beta.multi` yields the spatial turnover and the nestedness components of overall dissimilarity, and the sum of both components, total dissimilarity. The basic calculations for all these multiple-site measures and pairwise dissimilarity matrices can be computed using the function `betapart.core`, which returns an object of class `betapart`. This is useful for large datasets as the consuming calculations are done only once, and its result can then be used for computing many indices. The multiple-site values can be randomly sampled a specified number of times for a specified number of sites using the function `beta.sample`. The aforementioned indices used for assessing spatial patterns can also be used for measuring temporal changes in community composition with the function `beta.temp`. Likewise, an analogous framework has been implemented for separating the two components of abundance-based dissimilarity (balanced changes in abundance vs. abundance gradients) using commands `beta.pair.abund`, `beta.multi.abund`, `betapart.core.abund`, and `beta.sample.abund`. The framework has been extended for functional beta diversity with commands `functional.betapart.core`, `functional.beta.pair` and `functional.beta.multi`, and for phylogenetic beta diversity with commands `phylo.betapart.core`, `phylo.beta.pair` and `phylo.beta.multi`. The package also allows fitting negative exponential, power law or Gompertz distance-decay models for assessing the relationship between assemblage (dis)similarity and spatial (or other) distance. `decay.model` fits the nonlinear distance-decay function via the **minpack.lm** package, `plot.decay` plots the distance-decay pattern and the fitted model, `boot.coefs.decay` bootstraps the paramaters of the distance-decay model, and `zdep` assesses the differences between parameters of two distance-decay models.

Author(s)

Andres Baselga, David Orme, Sebastien Villéger, Julien De Bortoli, Fabien Leprieur, Maxime Logez, Sara Martínez-Santalla, Ramiro Martín-Devasa, Carola Gómez-Rodríguez, and Rosa M. Crujeiras

References

Baselga, A. 2010. Partitioning the turnover and nestedness components of beta diversity. Global Ecology and Biogeography 19:134-143

Baselga, A. 2012. The relationship between species replacement, dissimilarity derived from nestedness, and nestedness. Global Ecology and Biogeography 21, 1223-1232

Baselga, A. 2013. Separating the two components of abundance-based dissimilarity: balanced changes in abundance vs. abundance gradients. Methods in Ecology and Evolution, 4: 552-557

Baselga, A. 2017. Partitioning abundance-based multiple-site dissimilarity into components: balanced variation in abundance and abundance gradients. Methods in Ecology and Evolution 8: 799-808

Baselga A, Leprieur, F. 2015. Comparing methods to separate components of beta diversity. Methods in Ecology and Evolution 6: 1069-1079

Baselga A, Orme CDL. 2012. betapart: an R package for the study of beta diversity. Methods Ecol. Evol. 3: 808-812

Gómez-Rodríguez, C. & Baselga, A. 2018. Variation among European beetle taxa in patterns of distance decay of similarity suggests a major role of dispersal processes. Ecography, in press

Legendre P. 2014. Interpreting the replacement and richness difference components of beta diversity. Global Ecology and Biogeography, 23: 1324–1334

Leprieur F, Albouy C, De Bortoli J, Cowman PF, Belwood DR, Mouillot D. 2012. Quantifying phylogenetic beta diversity: distinguishing between "true" turnover of lineages and phylogenetic diversity gradients. PLoS One 7(8): e42760

Martín-Devasa R, Martínez-Santalla S, Gómez-Rodríguez C, Crujeiras RM, Baselga A. 2022. Species range size shapes distance decay in community similarity. Diversity and Distributions 28: 1348-1357

Martín-Devasa R, Martínez-Santalla S, Gómez-Rodríguez C, Crujeiras RM, Baselga A. 2022. Comparing distance-decay parameters: a novel test under pairwise dependence. Ecological Informatics 72: 101894

Martínez-Santalla S, Martín-Devasa R, Gómez-Rodríguez C, Crujeiras RM, Baselga A. 2022. Assessing the non-linear decay of community similarity: permutation and site-block resampling significance tests. Journal of Biogeography 49: 968-978

Villéger, S. Grenouillet, G., Brosse, S. 2013. Decomposing functional beta-diversity reveals that low functional beta-diversity is driven by low functional turnover in European fish assemblages. Global Ecology and Biogeography, 22: 671-681

`betapart.core` *Core calculations of betapart*

Description

Computes the basic quantities needed for computing the multiple-site beta diversity measures and pairwise dissimilarity matrices.

Usage

```
betapart.core(x)
```

Arguments

x	data frame, where rows are sites and columns are species

Value

The function returns an object of class `betapart` with the following elements:

sumSi	the sum of the species richness values of all sites
St	the total richness in the dataset
a	the multiple-site analog of the shared species term
shared	a matrix containing the number of species shared between pairs of sites
not.shared	a matrix containing the number of species not shared between pairs of sites: b, c
sum.not.shared	a matrix containing the total number of species not shared between pairs of sites: b+c
max.not.shared	a matrix containing the total maximum number of species not shared between pairs of sites: max(b,c)
min.not.shared	a matrix containing the total minimum number of species not shared between pairs of sites: min(b,c)

Author(s)

Andrés Baselga and David Orme

References

Baselga, A. 2010. Partitioning the turnover and nestedness components of beta diversity. Global Ecology and Biogeography 19:134-143

Baselga, A. 2012. The relationship between species replacement, dissimilarity derived from nestedness, and nestedness. Global Ecology and Biogeography 21, 1223-1232

See Also

beta.multi, beta.pair, beta.sample, beta.temp,

Examples

```
data(ceram.s)
ceram.core.s<-betapart.core(ceram.s)
ceram.dist.jac<-beta.pair(ceram.core.s, index.family="jac")
ceram.dist.sor<-beta.pair(ceram.core.s, index.family="sor")
ceram.multi.jac<-beta.multi(ceram.core.s, index.family="jac")
ceram.multi.sor<-beta.multi(ceram.core.s, index.family="sor")
```

`betapart.core.abund` *Core calculations of betapart for abundance-based dissimilarity measures*

Description

Computes the basic quantities needed for computing the abundance-based multiple-site dissimilarity measures and pairwise dissimilarity matrices.

Usage

```
betapart.core.abund(x)
```

Arguments

`x`	data frame, where rows are sites and columns are species

Value

The function returns an object of class `betapart.abund` with the following elements:

`multiple.shared.abund`
: the multiple-site intersection component in terms of abundances (AM)

`pair.shared.abund`
: a matrix containing the agreement in abundance between pairs of sites (A)

`min.not.shared.abund`
: a matrix containing the minimum disagreement in abundance between pairs of sites: min(B,C)

`max.not.shared.abund`
: a matrix containing the maximum disagreement in abundance between pairs of sites between pairs of sites: max(B,C)

`pair.not.shared.abund`
: a matrix containing the total disagreement in abundance between pairs of sites: B+C

Author(s)

Andrés Baselga

References

Baselga, A. 2013. Separating the two components of abundance-based dissimilarity: balanced changes in abundance vs. abundance gradients. Methods in Ecology and Evolution, 4: 552–557

Legendre, P. 2014. Interpreting the replacement and richness difference components of beta diversity. Global Ecology and Biogeography, 23: 1324–1334

Baselga, A. 2017. Partitioning abundance-based multiple-site dissimilarity into components: balanced variation in abundance and abundance gradients. Methods in Ecology and Evolution, 8: 799-808

See Also

beta.multi.abund, beta.pair.abund, beta.sample.abund, betapart.core

Examples

```
require(vegan)
data(BCI)
core.BCI<-betapart.core.abund(BCI)
pair.BCI<-beta.pair.abund(core.BCI)
multi.BCI<-beta.multi.abund(core.BCI)
```

---

betatest *A data set of 4 communities, 107 species and a 4D functional space*

Description

A data set to test the functions to dissimilarities matricies.

Usage

```
data("betatest")
```

Format

betatest is a list of two elements.

comm.test a dataframe with the presence/absence of 107 species among 4 sites

traits.test a dataframe of the traits (4 axes) of the 107 species

Examples

```
data(betatest)
str(betatest$comm.test)
str(betatest$traits.test)
```

---

boot.coefs.decay *Bootstrapping the parameters of distance-decay models computed with decay.model()*

Description

Takes the output of decay.model() and bootstraps the parameters of the model (i.e. intercept and slope in negative exponential or power law models, or position parameter and slope in Gompertz models).

Usage

```
boot.coefs.decay(m1, resamples, st.val = c(1, 0))
```

Arguments

m1	the output of decay.model().
resamples	the number of bootstrap resamples.
st.val	starting values for the nonlinear model.

Value

The function returns a list with:

model.type	functional form of the model, either negative exponential or power law.
y.type	similarties or dissimilarities.
boot.coefs	a matrix with the coefficients bootstrapped distributions, including values of the first parameter (intercept or position parameter) in the first column, and values of the second parameter (slope) in the second column.
original.coefs	model coefficients as estimated with a nonlinear model using decay.model().
mean.boot	the mean of the bootstrapped distributions.
sd.boot	the standard deviation of the bootstrapped distributions.

Author(s)

Sara Martínez-Santalla, Ramiro Martín-Devasa, Carola Gómez-Rodríguez, Rosa M. Crujeiras, Andrés Baselga

References

Gómez-Rodríguez, C. & Baselga, A. 2018. Variation among European beetle taxa in patterns of distance decay of similarity suggests a major role of dispersal processes. Ecography 41: 1825-1834

Martínez-Santalla S, Martín-Devasa R, Gómez-Rodríguez C, Crujeiras RM, Baselga A. 2022. Assessing the non-linear decay of community similarity: permutation and site-block resampling significance tests. Journal of Biogeography 49: 968-978

See Also

decay.model, zdep

Examples

```
# presence/absence tables for longhorn beetles of South Europe
data(ceram.s)

# spatial coordinates of territories in South Europe
data(coords.s)

# dissimilarity matrix
```

```
ceram.s.sim<-beta.pair(ceram.s)$beta.sim

# spatial distances in km
distgeo.s<-dist(coords.s[,1:2])

# Negative exponential model for the decay of similarity with spatial distance
decay.south<-decay.model(y=1-ceram.s.sim, x=distgeo.s, y.type="sim", model.type="exp")

# Site-block bootstrap
boot.coefs.decay(decay.south, resamples=100)
```

`bray.part` *Partitioning pair-wise Bray-Curtis dissimilarities*

Description

Computes 3 distance matrices accounting for the balanced variation and abundance gradient components of Bray-Curtis dissimilarity, and the sum of both values (i.e. Bray-Curtis dissimilarity)

Usage

```
bray.part(x)
```

Arguments

`x`	data frame of species abundances, where rows are sites and columns are species.

Value

The function returns a list with three dissimilarity matrices.

`bray.bal`	`dist` object, dissimilarity matrix accounting for the dissimilarity derived from balanced variation in abundance between sites
`bray.gra`	`dist` object, dissimilarity matrix accounting for the dissimilarity derived from unidirectional abundance gradients
`bray`	`dist` object, dissimilarity matrix accounting for total abundance-based dissimilarity between sites, measured as the Bray-Curtis index

Author(s)

Andrés Baselga

References

Baselga, A. 2013. Separating the two components of abundance-based dissimilarity: balanced changes in abundance vs. abundance gradients. Methods in Ecology and Evolution 4: 552–557

See Also

beta.pair

Examples

```
require(vegan)
data(BCI)
BCI.matrices<-bray.part(BCI)
```

---

ceram.n *Cerambycidae from Northern European Countries*

Description

The ceram.n data frame has 18 rows and 634 columns. Columns are presence/absence values of 634 species. The variable names are formed from the scientific names. The row names are standard country abbreviations, excepting RSS (Southern European Russia), RSC (Central European Russia) and RSN (Northern European Russia).

Usage

```
data(ceram.n)
```

Source

http://www.cerambycidae.net/

References

1. Danilevsky, M. L. 2007. A check-list of Longicorn Beetles (Coleoptera, Cerambycoidea) of Europe.Available at http://www.cerambycidae.net/
2. Baselga, A. 2008. Determinants of species richness, endemism and turnover in European longhorn beetles. Ecography 31:263-271

---

ceram.s *Cerambycidae from Southern European Countries*

Description

The ceram.s data frame has 15 rows and 634 columns. Columns are presence/absence values of 634 species. The variable names are formed from the scientific names. The case names are standard country abbreviations, excepting SS (Serbia) and CBH (Croatia and Bosnia-Herzegovina).

Usage

```
data(ceram.s)
```

Source

http://www.cerambycidae.net/

References

1. Danilevsky, M. L. 2007. A check-list of Longicorn Beetles (Coleoptera, Cerambycoidea) of Europe.Available at http://www.cerambycidae.net/
2. Baselga, A. 2008. Determinants of species richness, endemism and turnover in European longhorn beetles. Ecography 31:263-271

`coords.n` *Spatial coordinates for Southern European Countries*

Description

The coords.n data frame has 18 rows and 4 columns. Columns are UTM and latlong coordinates. The row names are standard country abbreviations, excepting RSS (Southern European Russia), RSC (Central European Russia) and RSN (Northern European Russia).

Usage

```
data(coords.n)
```

References

1. Baselga, A. 2008. Determinants of species richness, endemism and turnover in European longhorn beetles. Ecography 31:263-271

`coords.s` *Spatial coordinates for Southern European Countries*

Description

The coords.s data frame has 15 rows and 4 columns. Columns are UTM and latlong coordinates. TThe row names are standard country abbreviations, excepting SS (Serbia) and CBH (Croatia and Bosnia-Herzegovina).

Usage

```
data(coords.s)
```

References

1. Baselga, A. 2008. Determinants of species richness, endemism and turnover in European longhorn beetles. Ecography 31:263-271

`decay.model` *Fitting distance decay models to pair-wise assemblage similarity*

Description

Fits a nonlinear model describing (i) the decay of assemblage similarity with spatial (or any other) distance, or, equivalently, (ii) the increase of assemblage dissimilarity with distance. Nonlinear models are fitted via the nls.lm function in the minpack.lm package (which uses the Levenberg-Marquardt Nonlinear Least-Squares Algorithm). Implemented functional forms are either the (i) negative exponential, (ii) power law, or (iii) Gompertz models.

Usage

```
decay.model(y, x, model.type = "exponential", y.type = "similarities",
perm = 100, st.val = c(1, 0))
```

Arguments

`y`	`dist` object, either containing similarities or dissimilarities between pairs of assemblages.
`x`	`dist` object, containing distances (spatial or other) between pairs of assemblages.
`model.type`	functional form of the model, either negative exponential, power law, or Gompertz, partial match of `"exponential"`, `"power"`, or `"gompertz"`.
`y.type`	polarity of the `dist` object (i.e. 1 means total similarity or total dissimilarity), partial match of `"similarities"` or `"dissimilarities"`.
`perm`	number of permutations to assess significance.
`st.val`	starting values for the nonlinear model.

Value

The function returns a list with:

`data.y`	original y data, `dist` object, either containing similarities or dissimilarities between pairs of assemblages.
`data.x`	original x data, `dist` object, containing distances (spatial or other) between pairs of assemblages.
`model`	the fitted nonlinear model.
`model.type`	functional form of the model, either negative exponential, power law, or Gompertz.
`y.type`	similarties or dissimilarities.
`first.parameter`	first parameter of the model. It can be either the intercept, i.e. similarity or dissimilarity at distance=0, in negative exponential or power law models, or the position parameter in Gompertz models.

`second.parameter`
: slope of the model, i.e. rate at which similarity decreases with distance, or dissimilarity increases with distance in a negative exponential, power law or Gompertz model.

`aic`
: AIC of the model.

`pseudo.r.squared`
: proportion of the variation in the dependent variable that the model accounts for.

`p.value`
: significance of the model, as estimated from a permutation test.

Author(s)

Sara Martínez-Santalla, Ramiro Martín-Devasa, Carola Gómez-Rodríguez, Rosa M. Crujeiras, Andrés Baselga

References

Gómez-Rodríguez C, Baselga A. 2018. Variation among European beetle taxa in patterns of distance decay of similarity suggests a major role of dispersal processes. Ecography 41: 1825-1834

Martínez-Santalla S, Martín-Devasa R, Gómez-Rodríguez C, Crujeiras RM, Baselga A. 2022. Assessing the non-linear decay of community similarity: permutation and site-block resampling significance tests. Journal of Biogeography 49: 968-978

Martín-Devasa R, Martínez-Santalla S, Gómez-Rodríguez C, Crujeiras RM, Baselga A. 2022. Species range size shapes distance decay in community similarity. Diversity and Distributions 28: 1348-1357

See Also

`beta.pair`, `beta.pair.abund`, `plot.decay`, `boot.coefs.decay`, `zdep`

Examples

```
# presence/absence tables for longhorn beetles of South and North Europe
data(ceram.s)
data(ceram.n)

# spatial coordinates of territories in South and North Europe
data(coords.s)
data(coords.n)

# dissimilarity matrices
ceram.s.sim<-beta.pair(ceram.s)$beta.sim
ceram.n.sim<-beta.pair(ceram.n)$beta.sim

# spatial distances in km
distgeo.s<-dist(coords.s[,1:2])
distgeo.n<-dist(coords.n[,1:2])

# Negative exponential models for the decay of similarity with spatial distance
decay.south<-decay.model(y=1-ceram.s.sim, x=distgeo.s, y.type="sim", model.type="exp")
decay.north<-decay.model(y=1-ceram.n.sim, x=distgeo.n, y.type="sim", model.type="exp")
```

```
# Plot the decay models
plot.decay(decay.south, col="red")
plot.decay(decay.north, col="blue", add=TRUE)

# Equivalent models for the increase of dissimilarity with spatial distance
increase.south<-decay.model(y=ceram.s.sim, x=distgeo.s, y.type="dissim", model.type="exp")
increase.north<-decay.model(y=ceram.n.sim, x=distgeo.n, y.type="dissim", model.type="exp")

# Plot the decay models
plot.decay(increase.south, col="red")
plot.decay(increase.north, col="blue", add=TRUE)
```

`functional.beta.multi` *Multiple-site functional dissimilarities*

Description

Computes 3 multiple-site functional dissimilarities accounting for the spatial turnover and the nestedness components of functional beta diversity, and the sum of both values. Functional dissimilarities are based on volume of convex hulls intersections in a multidimensional functional space.

Usage

```
functional.beta.multi(x, traits, index.family="sorensen", warning.time=TRUE)
```

Arguments

`x`	data frame, where rows are sites and columns are species. Alternatively `x` can be a `functional.betapart` object derived from the `functional.betapart.core` function
`traits`	if `x` is not a `functional.betapart` object, a data frame, where rows are species and columns are functional space dimensions (i.e. quantitative traits or synthetic axes after PCoA). Number of species in each site must be strictly higher than number of dimensions. Number of dimensions should not exceed 4 and number of sites should not exceed 10. See `Details`.
`index.family`	family of dissimilarity indices, partial match of `"sorensen"` or `"jaccard"`.
`warning.time`	a logical value indicating whether computation of multiple-site dissimilarities would stop if number of dimensions exceeds 4 or if number of sites exceeds 10. If turn to `FALSE`, computation process can be tracked in the `step.fbc.txt` file, see `Details`.

Details

For multiple-site dissimilarities metrics (N>2 sites), the volume of the union of the N convex hulls is computed using the inclusion-exclusion principle (Villéger et al., 2011). It requires to compute the volume of all the intersections between 2 to N convex hulls. Intersection between k>2 convex hulls is computed as the intersection between the two convex hulls shaping intersections between the corresponding k-1 convex hulls, e.g. V(AnBnC)=V((AnB)n(BnC)). For N sites, computing multiple-site dissimilarity metrics thus requires computing 2^N-(N+1) pair-wise intersections between convex hulls in a multidimensional functional space. Computation time of the intersection between two convex hulls increases with the number of dimensions (D) of the functional space. Therefore, to prevent from running very long computation process `warning.time` is set by default to stop the function if D>4 or N>10. Computation progress can be tracked in the "step.fbc.txt" file written in the working directory. This table shows proportion of steps completed for computing convex hull volume shaping each site ("FRi") and intersections between them ("intersection_k"). Note that the the `betapart` package now supports external parallel computing for null models. However, this functionality is only availabe in `functional.betapart.core`. In this case, use the `functional.betapart` object as `x` in this function. See `functional.betapart.core` for more details.

Value

The function returns a list with the three multiple site functional dissimilarity values.

For `index.family="sorensen"` the three indices are:

`beta.SIM`	value of the functional turnover component, measured as Simpson derived functional dissimilarity
`beta.SNE`	value of the functional nestedness component, measured as nestedness-resultant fraction of Sorensen derived functional dissimilarity
`beta.SOR`	value of the overall functional beta diversity, measured as Sorensen derived functional dissimilarity

For `index.family="jaccard"` the three indices are:

`beta.JTU`	value of the functional turnover component, measured as turnover fraction of Jaccard derived functional dissimilarity
`beta.JNE`	value of the functional nestedness component, measured as nestedness-resultant fraction of Jaccard derived functional dissimilarity
`beta.JAC`	value of the overall functional beta diversity, measured as Jaccard derived functional dissimilarity

Author(s)

Sébastien Villéger, Andrés Baselga and David Orme

References

Villéger S., Novack-Gottshal P. & Mouillot D. 2011. The multidimensionality of the niche reveals functional diversity changes in benthic marine biotas across geological time. Ecology Letters 14: 561-568

Baselga, A. 2012. The relationship between species replacement, dissimilarity derived from nestedness, and nestedness. Global Ecology and Biogeography 21: 1223-1232

Villéger, S. Grenouillet, G., Brosse, S. 2013. Decomposing functional beta-diversity reveals that low functional beta-diversity is driven by low functional turnover in European fish assemblages. Global Ecology and Biogeography 22: 671–681

See Also

functional.beta.pair, functional.betapart.core, beta.multi

Examples

```
##### 4 communities in a 2D functional space (convex hulls are rectangles)
traits.test<-cbind( c(1,1,1,2,2,3,3,4,4,5,5) , c(1,2,4,1,2,3,5,1,4,3,5) )
dimnames(traits.test)<-list(paste("sp",1:11,sep="") , c("Trait 1","Trait 2") )

comm.test<-matrix(0,4,11,dimnames=list( c("A","B","C","D") , paste("sp",1:11,sep="") ) )
comm.test["A",c(1,2,4,5)]<-1
comm.test["B",c(1,3,8,9)]<-1
comm.test["C",c(6,7,10,11)]<-1
comm.test["D",c(2,4,7,9)]<-1

plot(5,5,xlim=c(0,6), ylim=c(0,6), type="n", xlab="Trait 1",ylab="Trait 2")
points(traits.test[,1],traits.test[,2], pch=21,cex=1.5,bg="black")
rect(1,1,4,4, col="#458B0050", border="#458B00") ; text(2.5,2.5,"B",col="#458B00",cex=1.5)

polygon(c(2,1,3,4), c(1,2,5,4), col="#DA70D650", border="#DA70D6") ;
text(2.5,3,"D",col="#DA70D6",cex=1.5)
rect(1,1,2,2, col="#FF000050" , border="#FF0000") ; text(1.5,1.5,"A",col="#FF0000",cex=1.5)

rect(3,3,5,5, col="#1E90FF50", border="#1E90FF") ; text(4,4.2,"C",col="#1E90FF",cex=1.5)


test.multi<-functional.beta.multi(x=comm.test, traits=traits.test, index.family = "jaccard" )
test.multi

test.multi.ABC<-functional.beta.multi(x=comm.test[c("A","B","C"),], traits=traits.test,
index.family = "jaccard" )
test.multi.ABC

test.multi.ABD<-functional.beta.multi(x=comm.test[c("A","B","D"),], traits=traits.test,
index.family = "jaccard" )
test.multi.ABD
```

---

functional.beta.pair *Pair-wise functional dissimilarities*

Description

Computes 3 distance matrices accounting for the spatial turnover and nestedness components of functional beta diversity, and the sum of both values. Functional dissimilarities are based on volume of convex hulls intersections in a multidimensional functional space.

Usage

```
functional.beta.pair(x, traits, index.family="sorensen")
```

Arguments

x	data frame, where rows are sites and columns are species. Alternatively x can be a functional.betapart object derived from the functional.betapart.core function or from the functional.betapart.core.pairwise function.
traits	if x is not a functional.betapart object, a data frame, where rows are species and columns are functional space dimensions (i.e. quantitative traits or synthetic axes after PCoA). Number of species in each site must be strictly higher than number of dimensions.
index.family	family of dissimilarity indices, partial match of "sorensen" or "jaccard".

Details

If x is a data.frame then functional.betapart.core.pairwise is called to compute the distance matrices necessary to compute the different components of the beta diversity. Only the default argument values will be used, while functional.betapart.core.pairwise integrates options that could be much more efficient, such as internal parallelisation, or different options for the convexhull volume estimation. Note that the the betapart package now supports external parallel computing for null models. As for internal parallelisation, these functionalities are only availabe in functional.betapart.core or in functional.betapart.core.pairwise. In this case, use the functional.betapart object as x in this function. See functional.betapart.core and functional.betepart.core.pairwise for more details.

Value

The function returns a list with three functional dissimilarity matrices.

For index.family="sorensen" the three matrices are:

funct.beta.sim	dist object, dissimilarity matrix accounting for functional turnover, measured as Simpson derived pair-wise functional dissimilarity
funct.beta.sne	dist object, dissimilarity matrix accounting for nestedness-resultant functional dissimilarity, measured as the nestedness-fraction of Sorensen derived pair-wise functional dissimilarity
funct.beta.sor	dist object, dissimilarity matrix accounting for functional beta diversity, measured as Sorensen derived pair-wise functional dissimilarity

For index.family="jaccard" the three matrices are:

funct.beta.jtu	dist object, dissimilarity matrix accounting for functional turnover, measured as the turnover-fraction of Jaccard derived pair-wise functional dissimilarity

`funct.beta.jne` `dist` object, dissimilarity matrix accounting for nestedness-resultant functional dissimilarity, measured as the nestedness-fraction of Jaccard derived pair-wise functional dissimilarity

`funct.beta.jac` `dist` object, dissimilarity matrix accounting for functional beta diversity, measured as Jaccard derived pair-wise functional dissimilarity

Author(s)

Sébastien Villéger, Andrés Baselga and David Orme

References

Villéger S., Novack-Gottshal P. & Mouillot D. 2011. The multidimensionality of the niche reveals functional diversity changes in benthic marine biotas across geological time. Ecology Letters 14: 561-568

Baselga, A. 2012. The relationship between species replacement, dissimilarity derived from nestedness, and nestedness. Global Ecology and Biogeography 21: 1223-1232

Villéger, S. Grenouillet, G., Brosse, S. 2013. Decomposing functional beta-diversity reveals that low functional beta-diversity is driven by low functional turnover in European fish assemblages. Global Ecology and Biogeography 22: 671–681

See Also

`functional.beta.multi`, `functional.betapart.core`, `functional.betapart.core.pairwise`, `beta.pair`

Examples

```
##### 4 communities in a 2D functional space (convex hulls are rectangles)
traits.test<-cbind( c(1,1,1,2,2,3,3,4,4,5,5) , c(1,2,4,1,2,3,5,1,4,3,5) )
dimnames(traits.test)<-list(paste("sp",1:11,sep="") , c("Trait 1","Trait 2") )

comm.test<-matrix(0,4,11,dimnames=list( c("A","B","C","D") , paste("sp",1:11,sep="") ) )
comm.test["A",c(1,2,4,5)]<-1
comm.test["B",c(1,3,8,9)]<-1
comm.test["C",c(6,7,10,11)]<-1
comm.test["D",c(2,4,7,9)]<-1

plot(5,5,xlim=c(0,6), ylim=c(0,6), type="n", xlab="Trait 1",ylab="Trait 2")
points(traits.test[,1],traits.test[,2], pch=21,cex=1.5,bg="black")
rect(1,1,4,4, col="#458B0050", border="#458B00") ; text(2.5,2.5,"B",col="#458B00",cex=1.5)

polygon(c(2,1,3,4), c(1,2,5,4), col="#DA70D650", border="#DA70D6") ;
text(2.5,3,"D",col="#DA70D6",cex=1.5)
rect(1,1,2,2, col="#FF000050" , border="#FF0000") ; text(1.5,1.5,"A",col="#FF0000",cex=1.5)

rect(3,3,5,5, col="#1E90FF50", border="#1E90FF") ; text(4,4.2,"C",col="#1E90FF",cex=1.5)


test.pair<-functional.beta.pair(x=comm.test, traits=traits.test, index.family = "jaccard")
```

```
lapply(test.pair,round,2)

#### with functional.betapart.core.pairwise
test1 <- functional.betapart.core.pairwise(comm.test, traits.test)
test.pair <- functional.beta.pair(test1)
## Not run:
#### if internal parallelisation would be interesting (large community matrix)
test1 <- functional.betapart.core.pairwise(comm.test, traits.test, parallel = TRUE,
                                           opt.parallel = list(nc = 2))
test.pair <- functional.beta.pair(test1)

## End(Not run)
```

`functional.betapart.core`

Core calculations of functional dissimilarities metrics

Description

Computes the basic quantities needed for computing the multiple-site functional beta diversity measures and pairwise functional dissimilarity matrices. This version of the function now supports internal parallelization to fasten the computations and external parallelization for null models.

Usage

```
functional.betapart.core(x, traits, multi = TRUE, warning.time = TRUE,
                         return.details = FALSE, fbc.step = FALSE,
                         parallel = FALSE, opt.parallel = beta.para.control(),
                         convhull.opt = qhull.opt(),
                         progress = FALSE)
```

Arguments

`x`	data frame, where rows are sites and columns are species.
`traits`	data frame, where rows are species and columns are functional space dimensions (i.e. quantitative traits or synthetic axes after PCoA). Number of species in each site must be strictly higher than number of dimensions.
`multi`	a logical value indicating whether basic quantities for multiple-site functional beta-diversity should be computed. See `Details`.
`warning.time`	a logical value indicating whether computation of multiple-site dissimilarities would stop if number of dimensions exceeds 4 or if number of sites exceeds 10. If turn to `FALSE`, computation process can be tracked in the `step.fbc.txt` file, see `Details`.
`return.details`	a logical value indicating whether volume and coordinates of vertices of convex hulls shaping each site and their intersections in the functional space should be returned.

fbc.step	a logical value indicating whether the computation progress tracking file "step.fbc.txt" should be created; Setting it to FALSE will speed up the function. It is automatically turn to FALSE when parallel is TRUE.
parallel	a logical value indicating if internal parallelization is used to compute pairwise dissimilarities, see Examples. If multi is set to TRUE parallelization will be turned-off.
opt.parallel	a list of values to replace the default values returned by the function beta.para.control to customize the cluster used for parallel computing.
convhull.opt	a list of values to replace the default values returned by the function qhull.opt, to pass options to the convhulln function. The first element, conv1, set the options that be used by default with convhulln, while the second element, conv2, set the options that will be used if convhulln returns an error. By default (conv1 = 'QJ' and conv2 = NULL). If convhulln generates an error, a NA is returned. This prevents the function to stop when encountering an error.
progress	a logical indicating if a progress bar should be displayed (TRUE) or not (by default).

Details

For multiple-site dissimilarities metrics (N>2 sites), the volume of the union of the N convex hulls is computed using the inclusion-exclusion principle (Villéger et al., 2011). It requires to compute the volume of all the intersections between 2 to N convex hulls. Intersection between k>2 convex hulls is computed as the intersection between the two convex hulls shaping intersections between the corresponding k-1 convex hulls, e.g. V(AnBnC)=V((AnB)n(BnC)). For N sites, computing multiple-site dissimilarity metrics thus requires computing 2^N-(N+1) pair-wise intersections between convex hulls in a multidimensional functional space. Computation time of the intersection between two convex hulls increases with the number of dimensions (D) of the functional space. Therefore, to prevent from running very long computation process warning.time is set by default to stop the function if D>4 or N>10.

If fbc.step is set to TRUE, computation progress can be tracked in the "step.fbc.txt" file written in the working directory. This table shows proportion of steps completed for computing convex hull volume shaping each site ("FRi") and intersections between them ("intersection_k"). This is only possible when computations are not performed in parallel, and this whatever the type of parallelization used (external or internal).

If parallel is set to TRUE, computation will be run though the creation of a cluster. This is interesting when beta diversity computation is long. When the number of sites increase and/or when the taxonomic richness is highly variable between sites, parallelization becomes more and more interesting. On small matrices, the running time could inflate due to the creation of the cluster and its management.

Value

The function returns an object of class betapart with the following elements:

sumFRi	the sum of the functional richness values of all sites
FRt	the total functional richness in the dataset
a	the multiple-site analog of the shared functional richness term

shared	a matrix containing the functional richness shared between pairs of sites
not.shared	a matrix containing the functional richness not shared between pairs of sites: b, c
sum.not.shared	a matrix containing the total functional richness not shared between pairs of sites: b+c
max.not.shared	a matrix containing the total maximum functional richness not shared between pairs of sites: max(b,c)
min.not.shared	a matrix containing the total minimum functional richness not shared between pairs of sites: min(b,c)
details	if return.details=TRUE a list of two lists: $CH a list with a vector (FRi) of functional richness in each site (i.e. convex hull volume) and coord_vertices a list of N matrices with the coordinates of species being vertices in the D-dimensions functional space. $intersections a list of 3 lists: $combinations, N-1 matrices with all combinations of 2 to N sites (numbers are rank of sites in x) ; $volumes, N-1 vectors with the volume inside the intersection between each combination of sites ; $coord_vertices, list of N-1 matrices with the coordinates of the vertices shaping these intersections (NA if no intersection). See http://www.qhull.org/html/qh-optq.htm for the possible options.

Author(s)

Sébastien Villéger, Andrés Baselga, David Orme, Renato Henriques-Silva, Maxime Logez

References

Villéger S., Novack-Gottshal P. & Mouillot D. 2011. The multidimensionality of the niche reveals functional diversity changes in benthic marine biotas across geological time. Ecology Letters. 14, 561-568

Baselga, A. 2012. The relationship between species replacement, dissimilarity derived from nestedness, and nestedness. Global Ecology and Biogeography 21, 1223-1232

Villéger, S. Grenouillet, G., Brosse, S. 2012. Decomposing functional beta-diversity reveals that low functional beta-diversity is driven by low functional turnover in European fish assemblages. Global Ecology and Biogeography, in press

See Also

functional.beta.multi, functional.beta.pair, betapart.core

Examples

```
##### 4 communities in a 2D functional space (convex hulls are rectangles)
traits.test <- cbind(c(1, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5),
                     c(1, 2, 4, 1, 2, 3, 5, 1, 4, 3, 5))
dimnames(traits.test) <- list(paste("sp", 1:11, sep=""), c("Trait 1", "Trait 2"))

comm.test <- matrix(0, 4, 11, dimnames = list(c("A", "B", "C", "D"),
                    paste("sp", 1:11, sep="")))
comm.test["A", c(1, 2, 4, 5)] <- 1
```

```
comm.test["B", c(1, 3, 8, 9)] <- 1
comm.test["C", c(6, 7, 10, 11)] <- 1
comm.test["D", c(2, 4, 7, 9)] <- 1

plot(5, 5, xlim = c(0, 6), ylim = c(0, 6), type = "n", xlab = "Trait 1", ylab = "Trait 2")
points(traits.test[, 1], traits.test[, 2], pch = 21, cex = 1.5, bg = "black")
rect(1, 1, 4, 4, col = "#458B0050", border = "#458B00")
text(2.5, 2.5, "B" , col = "#458B00", cex = 1.5)
polygon(c(2, 1, 3, 4), c(1, 2, 5, 4), col = "#DA70D650", border = "#DA70D6")
text(2.5, 3, "D", col = "#DA70D6", cex = 1.5)
rect(1, 1, 2, 2, col = "#FF000050", border = "#FF0000")
text(1.5, 1.5, "A", col = "#FF0000", cex = 1.5)
rect(3, 3, 5, 5, col = "#1E90FF50", border = "#1E90FF")
text(4, 4.2, "C", col = "#1E90FF", cex = 1.5)

# for multiple dissimilarity, multi = TRUE
test.core <- functional.betapart.core(x = comm.test, traits = traits.test,
                                      multi = TRUE, return.details = FALSE)
test.core

# for pairwise dissimilarity, multi = FALSE
test.core <- functional.betapart.core(x = comm.test, traits = traits.test,
                                      multi = FALSE, return.details = FALSE)
test.core

# to use systematilcally the "QJ" options
test.core <- functional.betapart.core(x = comm.test, traits = traits.test,
                                      multi = FALSE, return.details = FALSE,
                                      convhull.opt = list(conv1 = "QJ"))

# to use the "QJ" options only if the convhull function generates an error
# instead of returning NA
test.core <- functional.betapart.core(x = comm.test, traits = traits.test,
                                      multi = FALSE, return.details = FALSE,
                                      convhull.opt = list(conv2 = "QJ"))

# using functional.betapart.core to get details on intersections
# when only pairwise dissimilarity is computed
test.core.pair <- functional.betapart.core(x = comm.test, traits = traits.test,
                                           multi = FALSE, return.details = TRUE)
test.core.pair

# for multiple dissimilarity
test.core.multi <- functional.betapart.core(x = comm.test, traits = traits.test,
                                            multi = TRUE, return.details = TRUE)
test.core.multi


# using core outputs to compute pairwise and multiple functional dissimilarities
functional.beta.pair(x = test.core.pair, index.family = "jaccard" )
functional.beta.multi(x = test.core.multi, index.family = "jaccard" )
```

```
##### using internal parallelisation to fasten pairiwse dissimilarity
# by default it use serial computation
test.core.pair <- functional.betapart.core(x = comm.test, traits = traits.test,
                                           multi = FALSE, return.details = FALSE,
                                           fbc.step = FALSE, parallel = FALSE)

# by default it uses half of the cores and 1 task per run (this can be customised)
# test.core.pairp <- functional.betapart.core(x = comm.test, traits = traits.test,
#                                             multi = FALSE, return.details = FALSE,
#                                             fbc.step = FALSE, parallel = TRUE)

# you can set the number of core to use :
test.core.pairp <- functional.betapart.core(x = comm.test, traits = traits.test,
                                            multi = FALSE, return.details = FALSE,
                                            fbc.step = FALSE, parallel = TRUE,
                                            opt.parallel = beta.para.control(nc = 2))
all.equal(test.core.pair, test.core.pairp)

# library(microbenchmark)
# microbenchmark(serial =
#                  functional.betapart.core(comm.test, traits.test, multi = FALSE,
#                                           return.details = FALSE, fbc.step = FALSE,
#                                           parallel = FALSE),
#                nc2 =
#                  functional.betapart.core(comm.test, traits.test, multi = FALSE,
#                                           return.details = FALSE, fbc.step = FALSE,
#                                           parallel = TRUE,
#                                           opt.parallel = beta.para.control(nc = 2)),
#                nc4 =
#                  functional.betapart.core(comm.test, traits.test, multi = FALSE,
#                                           return.details = FALSE, fbc.step = FALSE,
#                                           parallel = TRUE,
#                                           opt.parallel = beta.para.control(nc = 4))
# )

## Not run:
# If the number of species is very different among communities
# load-balancing parallelisation could be very efficient
# especially when the number of community is high
test.core.pairp <- functional.betapart.core(comm.test, traits.test, multi = FALSE,
                                            return.details = FALSE, fbc.step = FALSE,
                                            parallel = TRUE,
                                            opt.parallel =
                                              beta.para.control(nc = 2, LB = TRUE))

# use you can use fork cluster (but not on Windows)
#test.core.pairp <- functional.betapart.core(comm.test, traits.test, multi = FALSE,
#                                            return.details = FALSE, fbc.step = FALSE,
#                                            parallel = TRUE,
#                                            opt.parallel =
#                                              beta.para.control(nc = 2, type = "FORK"))

# finally you can customise the number of task run at each time
```

```
test.core.pairp <- functional.betapart.core(comm.test, traits.test, multi = FALSE,
                                            return.details = FALSE, fbc.step = FALSE,
                                            parallel = TRUE,
                                            opt.parallel =
                                              beta.para.control(nc = 2, size = 6))

# using internal parallelisation is not always useful, especially on small data set
# load balancing is very helpful when species richness are highly variable

# Null model using 'external' parallel computing

# Example 1: pairwise functional beta diversity (functional.beta.pair)
# Note that this is an example with a small number of samples and null model
# permutations for illustration.
# Real null model analyses should have a much greater number of samples and permutations.

##### 4 communities in a 3D functional space

comm.test <- matrix(0, 4, 11, dimnames = list(c("A", "B", "C", "D"),
                                              paste("sp", 1:11, sep = "")))
comm.test["A", c(1, 2, 4, 5)] <- 1
comm.test["B", c(1, 3, 8, 9)] <- 1
comm.test["C", c(6, 7, 10, 11)] <- 1
comm.test["D", c( 2, 4, 7, 9)] <- 1

set.seed(1)
traits.test <- matrix(rnorm(11*3, mean = 0, sd = 1), 11, 3)
dimnames(traits.test) <- list(paste("sp", 1:11, sep = "") ,
                              c("Trait 1", "Trait 2", "Trait 3"))

# Required packages
library(doSNOW)
library(picante)
library(fastmatch)
library(foreach)


# define number of cores
# Use parallel::detectCores() to determine number of cores available in your machine
nc <- 2

# 4 cores would be better (nc <- 4)

# create cluster
cl <- snow::makeCluster(nc)

# register parallel backend
doSNOW::registerDoSNOW(cl)

# define number of permutations for the null model (the usual is 1000)
# make sure that nperm/nc is a whole number so that all cores have the same number
# of permutations to work on
nperm <- 100
```

```
test.score <- functional.betapart.core(comm.test, traits.test, multi = FALSE,
                                       warning.time = FALSE, return.details = FALSE,
                                       fbc.step = FALSE, parallel = FALSE)

obs.pair.func.dis <- functional.beta.pair(x = test.score, index.family = "sorensen")

# transform functional.beta.pair results into a matrix
obs.pair.func.dis <- do.call(rbind, obs.pair.func.dis)

# set names for each pair of site
pair_names <- combn(rownames(comm.test), 2, FUN = paste, collapse = "_")
colnames(obs.pair.func.dis) <- pair_names

# export necessary variables and functions to the cluster of cores
snow::clusterExport(cl = cl, c("comm.test", "traits.test"),
                        envir = environment())

# creation of an iterator to run 1 comparaisons on each core at time
it <- itertools::isplitIndices(nperm, chunkSize = 1)

# parallel computation
null.pair.func.dis <-
  foreach(n = it, .combine = c, .packages=c("picante","betapart","fastmatch")) %dopar% {

          # it enables to adjust the number of permutations (nt) done on each run
          nt <- length(n)
          null.pair.temp <- vector(mode = "list", length = nt)

          # for each core "n" perform "nt" permutations
          for (j in 1:nt){

            # randomize community with chosen null model
            # for this particular example we used the "independent swap algorithm"
            # but the user can choose other types of permutation
            # or create it's own null model
            null.comm.test <- randomizeMatrix(comm.test, null.model = "independentswap",
                                              iterations=1000)

            # execute functional.betapart.core function
            null.test.score <-
              try(functional.betapart.core(null.comm.test, traits = traits.test,
                                           multi = FALSE, warning.time = FALSE,
                                           return.details = FALSE,  fbc.step = FALSE,
                                           parallel = FALSE), silent = TRUE)
            # using 'external' parallelisation it is necessary to set parralel to FALSE

            # in this artificial example there are a few combinations of species that
            # the convex hull cannot be calculated due to some odd geometric combination
            # so we need to re-permute the community matrix
            while(inherits(null.test.score, "try-error")){

             null.comm.test <- randomizeMatrix(comm.test, null.model = "independentswap",
```

```
                                              iterations = 1000)
              null.test.score <-
                try(functional.betapart.core(x = null.comm.test, traits = traits.test,
                                             multi = FALSE, warning.time = FALSE,
                                             return.details = FALSE, fbc.step = FALSE,
                                             parallel = FALSE), silent = TRUE)
              }

            # compute the pairwise beta-diversity null values and input them in the
            # temporary result matrix
            res <- functional.beta.pair(x = null.test.score, index.family = "sorensen")
            null.pair.temp[[j]] <- do.call(rbind, res)

        }
        #retrieve the results from each core
        null.pair.temp
    }

# stop cluster
snow::stopCluster(cl)


# compute the mean, standard deviation and p-values of dissimilarity metrics
# for each pair of site

mean.null.pair.func <- matrix(numeric(), ncol = ncol(obs.pair.func.dis),
                              nrow = nrow(obs.pair.func.dis))
sd.null.pair.func <- matrix(numeric(), ncol = ncol(obs.pair.func.dis),
                            nrow = nrow(obs.pair.func.dis))
p.pair.func.dis <- matrix(numeric(), ncol = ncol(obs.pair.func.dis),
                          nrow = nrow(obs.pair.func.dis))

# for each one of the 3 null dissimilarity metrics (SIN, SNE and SOR)
for (j in 1:nrow(obs.pair.func.dis)){
  matnull <- sapply(null.pair.func.dis, function(x) x[j,])
  mean.null.pair.func[j,] <- rowMeans(matnull)
  sd.null.pair.func[j,] <- sqrt(rowSums((matnull - mean.null.pair.func[j,])^2)/(nperm-1))
  p.pair.func.dis[j,] <- rowSums(matnull >= obs.pair.func.dis[j,])
 p.pair.func.dis[j,] <- (pmin(p.pair.func.dis[j,],nperm-p.pair.func.dis[j,])+1)/(nperm+1)
  # the +1 is to take into account that the observed value is one of the possibilities
}

# compute standardized effect sizes
ses.pair.func.dis <- (obs.pair.func.dis - mean.null.pair.func)/sd.null.pair.func


# Example 2: multiple functional beta diversity (functional.beta.multi)
# Note that this is an example with a small number of samples and null model
# permutations for illustration.
# Real null model analyses should have a much greater number of samples
# and permutations.

##### 4 communities in a 3D functional space
```

```
comm.test <- matrix(0, 4, 11,dimnames = list(c("A", "B", "C", "D"),
                                             paste("sp", 1:11, sep = "")))
comm.test["A", c(1, 2, 4, 5)] <- 1
comm.test["B", c(1, 3, 8, 9)] <- 1
comm.test["C", c(6, 7, 10, 11)] <- 1
comm.test["D", c(2, 4, 7, 9)]<-1

set.seed(1)
traits.test <- matrix(rnorm(11*3, mean=0, sd=1), 11, 3)
dimnames(traits.test) <- list(paste("sp", 1:11, sep=""),
                              c("Trait 1", "Trait 2", "Trait 3"))

# Required packages
library(doSNOW)
library(picante)
library(fastmatch)
library(foreach)


# define number of cores
# Use parallel::detectCores() to determine number of cores available in your machine
nc <- 2

# create cluster
cl <- snow::makeCluster(nc)

# register parallel backend
doSNOW::registerDoSNOW(cl)

# define number of permutations for the null model (the usual is 1000)
# make sure that nperm/nc is a whole number so that all cores have the same number
# of permutations to work on
nperm <- 10

# compute observed values for multiple functional dissimilarities
test.score <- functional.betapart.core(comm.test, traits.test, multi = TRUE,
                                       warning.time = FALSE, return.details = FALSE,
                                       fbc.step = FALSE, parallel = FALSE)
obs.multi.func.dis <- do.call(cbind, functional.beta.multi(test.score,
                                                           index.family = "sorensen"))

# export necessary variables and functions to the cluster of cores
snow::clusterExport(cl = cl, c("comm.test", "traits.test"),
                        envir=environment())

it <- itertools::isplitIndices(nperm, chunkSize = 1)

null.multi.func.dis <-
  foreach(n = it, .combine = rbind,
          .packages = c("picante","betapart","fastmatch")) %dopar% {

    # for each core, create temporary matrix to store 3 null multiple functional
```

```
    # dissimilarity indices (SIN, SNE,SOR)
    null.multi.temp <- matrix(numeric(), ncol = 3, nrow = length(n),
                              dimnames = list(NULL, c("funct.beta.SIM", "funct.beta.SNE",
                                                      "funct.beta.SOR")))

    # number of tasks per core (i.e., permutations per core)
    nt <- length(n)

    # for each core "n" perform "nt" permutations
    for (j in 1:nt) {

      # randomize community matrix with chosen null model (for this example
      # we chose the "independent swap" algorithm)
      null.comm.test <- randomizeMatrix(comm.test, null.model="independentswap",
                                        iterations=1000)

      # execute functional.betapart.core function identifying each "n" core
      # with the core.ident argument for external parallelization,
      null.test.score <-
        try(functional.betapart.core(null.comm.test, traits.test, multi = TRUE,
                                     warning.time = FALSE, return.details = FALSE,
                                     fbc.step = FALSE, parallel = FALSE),
            silent = TRUE)

      # in this artificial example there are a few combinations of species
      # that the convex hull
      # cannot be calculated due to some odd geometric combination so we
      # need to re-permute the community matrix

      while(inherits(null.test.score, "try-error")){
        null.comm.test <- randomizeMatrix(comm.test, null.model="independentswap",
                                          iterations=1000)
        null.test.score <-
          try(functional.betapart.core(null.comm.test, traits.test, multi = TRUE,
                                       warning.time = FALSE, return.details = FALSE,
                                       fbc.step = FALSE, parallel = FALSE),
              silent = TRUE)
        }
      # input null values in the temporary result matrix
      null.multi.temp[j,] <- unlist(functional.beta.multi(null.test.score,
                                                          index.family = "sorensen"))
      }
    # recover results from each core
    null.multi.temp
    }

# close cluster
snow::stopCluster(cl)

# result matrix
result <- matrix(numeric(), ncol = 3, nrow = 3,
                 dimnames = list(c("obs", "ses", "p"), colnames(obs.multi.func.dis)))
```

```
# input observed values for the multiple functional dissimilarity indices (SIN, SNE,SOR)
result[1,] = obs.multi.func.dis

# compute standardized effect sizes (ses) for the multiple functional
# dissimilarity indices (SIN, SNE,SOR)
result[2,] <- (obs.multi.func.dis-colMeans(null.multi.func.dis, na.rm=TRUE))/
  apply(null.multi.func.dis,2, sd, na.rm=TRUE)

# compute p-values for the multiple functional dissimilarity indices (SIN, SNE,SOR)
for (i in 1:3) {
  result[3, i] <- sum(obs.multi.func.dis[i]<=null.multi.func.dis[,i])
  result[3, i] <- (pmin(result[3, i], nperm - result[3, i]) + 1)/(nperm+1)
}
# the +1 is to take into account that the observed value is one of the possibilities

result
###

## End(Not run)
```

`functional.betapart.core.pairwise`

functional.betapart.core.pairwise

Description

Computes the basic quantities needed for computing the pairwise functional dissimilarity matrices. This function is similar to functional.betapart.core with multi=FALSE but it provides more options for computing convex hulls shaping each assemblage (through option passed to qhull algorithm in 'geometry::convhulln()' as well as their intersections (computed based on library 'geometry' whenever possible, else with 'rcdd' as in functional.betapart.core.

Usage

```
functional.betapart.core.pairwise(x, traits,
                                  return.details = TRUE,
                                  parallel = FALSE,
                                  opt.parallel = beta.para.control(),
                                  convhull.opt = qhull.opt(),
                                  progress = FALSE)
```

Arguments

x	A data frame, where rows are sites and columns are species.
traits	A data frame, where rows are species and columns are functional space dimensions (i.e. quantitative traits or synthetic axes after PCoA). Number of species in each site must be strictly higher than number of dimensions.

`return.details` A logical value indicating if informations concerning the computation of the convexhull volumes should be returned (`TRUE`) or not (`FALSE`). See `Details`.

`parallel` A logical value indicating if internal parallelization is used to compute pairwise dissimilarities (`TRUE`), see Examples..

`opt.parallel` A list of four values to modify default values used to define and run the parallel cluster. See `beta.para.control` and the `Details` section.

`convhull.opt` A list of two named vectors of character `conv1` and `conv2` defining the options that will be passed to the `convhulln` function to compute the convexhull volumes (`http://www.qhull.org/html/qh-optq.htm`). `conv1` sets the default option that will be passed tp `convhulln` (`'QJ'` by default instead of `'Qt'`), while `conv2` set the options if `convhulln` return an error with options set by `conv1`. See `Details` and the examples.

`progress` A logical indicating if a progress bar should be displayed (`TRUE`) or not (by default) (`FALSE`).

Details

opt.parallel Among the four options (see `beta.para.control`), the number of cores (`nc`) and the number of convexhull volumes computed by each core at each iteration (`size`) are the most important with this function. As `inter_rcdd` is very fast, it is necessary to set a large value (>100) for `size`. Otherwise the parallelisation would not be so efficient. With a low number of communities, using internal parallelisation will slow down the function.

convhull.opt Some specific distributions of points could generate errors when computing the convexhull volumes with the default qhull options ('Qt'), which is why `'QJ'` was preferred as the default value (`conv1`). Sometimes, it could be interesting to use alternative options such as 'Qt' or 'Qs' to achieve the computation. These alternative options could be used to compute all the convexhull volumes with `conv1`, or only when an error occurs using `conv2`. It is thus possible to define `conv1` as `'Qt'` to use the default 'qhull' options, but to prevent errors by setting `'QJ'` to the `conv2` argument.

Value

The function returns an object of class `betapart` with the following elements:

sumFRi The sum of the functional richness values of all sites

FRt The total functional richness in the dataset `NA`. Kept for compatibility with `functional.betapart.core`

a The multiple-site analog of the shared functional richness term, `NA`. Kept for compatibility with `functional.betapart.core`

shared A matrix containing the functional richness shared between pairs of sites

not.shared A matrix containing the functional richness not shared between pairs of sites: b, c

sum.not.shared A matrix containing the total functional richness not shared between pairs of sites: b+c

max.not.shared A matrix containing the total maximum functional richness not shared between pairs of sites: max(b,c)

min.not.shared A matrix containing the total minimum functional richness not shared between pairs of sites: min(b,c)

details NA if return.details = FALSE. Otherwise a list of two elements:

$FRi A data frame with two columns: the FRi values and the qhull options used to compute them (qhull.opt).

$Intersection A data frame with the pairs of communities (Comms), the function used to compute the volume of their intersections (Inter), and the qhull options used (qhull.opt).

Examples

```
##### 4 communities in a 2D functional space (convex hulls are rectangles)
traits.test <- cbind(c(1, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5),
                     c(1, 2, 4, 1, 2, 3, 5, 1, 4, 3, 5))
dimnames(traits.test) <- list(paste("sp", 1:11, sep=""), c("Trait 1", "Trait 2"))

comm.test <- matrix(0, 4, 11, dimnames = list(c("A", "B", "C", "D"),
                                              paste("sp", 1:11, sep="")))
comm.test["A", c(1, 2, 4, 5)] <- 1
comm.test["B", c(1, 3, 8, 9)] <- 1
comm.test["C", c(6, 7, 10, 11)] <- 1
comm.test["D", c(2, 4, 7, 9)] <- 1

plot(5, 5, xlim = c(0, 6), ylim = c(0, 6), type = "n", xlab = "Trait 1", ylab = "Trait 2")
points(traits.test[, 1], traits.test[, 2], pch = 21, cex = 1.5, bg = "black")
rect(1, 1, 4, 4, col = "#458B0050", border = "#458B00")
text(2.5, 2.5, "B" , col = "#458B00", cex = 1.5)
polygon(c(2, 1, 3, 4), c(1, 2, 5, 4), col = "#DA70D650", border = "#DA70D6")
text(2.5, 3, "D", col = "#DA70D6", cex = 1.5)
rect(1, 1, 2, 2, col = "#FF000050", border = "#FF0000")
text(1.5, 1.5, "A", col = "#FF0000", cex = 1.5)
rect(3, 3, 5, 5, col = "#1E90FF50", border = "#1E90FF")
text(4, 4.2, "C", col = "#1E90FF", cex = 1.5)

# for pairwise dissimilarity
test.core <- functional.betapart.core.pairwise(x = comm.test, traits = traits.test,
                                               return.details = FALSE)
test.core
# equivalent to
test <- functional.betapart.core(x = comm.test, traits = traits.test,
                                 return.details = FALSE,
                                 multi = FALSE)
all.equal(test.core, test)

# using core outputs to compute pairwise and multiple functional dissimilarities
functional.beta.pair(x = test.core, index.family = "jaccard" )

## Not run:
#### using convhulln options
# a data set that generates NA (due to errors) with functional.betapart.core
data(betatest)
# a list of 2 data.frame : comm.test & traits.test
comm.test <- betatest$comm.test
traits.test <- betatest$traits.test
```

```
test <- functional.betapart.core(x = comm.test, traits = traits.test,
                                 return.details = FALSE,
                                 multi = FALSE)

any(is.na(test$shared))
# no NA because the default option was set to QJ
# if we use the default option of qhull (Qt) :
test <- functional.betapart.core(x = comm.test, traits = traits.test,
                                 return.details = FALSE,
                                 multi = FALSE,
                                 convhull.opt = list(conv1 = "Qt"))
any(is.na(test$shared))
# some NA arise

## End(Not run)
# with functional.betapart.core.pairwise
test.core <- functional.betapart.core.pairwise(x = comm.test, traits = traits.test,
                                               return.details = FALSE,
                                               convhull.opt = list(conv1 = "Qt"))
any(is.na(test.core$shared))
# here no NA were generated because the volumes of the intersections
# were computed only with inter_geom
# to know which functions were used, set return.details to TRUE
test.core <- functional.betapart.core.pairwise(x = comm.test, traits = traits.test,
                                               return.details = TRUE,
                                               convhull.opt = list(conv1 = "Qt"))
test.core$details$Intersection

#### convhull options
traits.test <- cbind(c(1, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5) ,
                     c(1, 2, 4, 1, 2, 3, 5, 1, 4, 3, 5))
dimnames(traits.test) <- list(paste("sp", 1:11, sep=""), c("Trait 1", "Trait 2"))

comm.test <- matrix(0, 4, 11,
                    dimnames = list(c("A", "B", "C", "D"), paste("sp", 1:11, sep="")))
comm.test["A", c(1, 2, 4, 5)] <- 1
comm.test["B", c(1, 3, 8, 9)] <- 1
comm.test["C", c(6, 7, 10, 11)] <- 1
comm.test["D", c(2, 4, 7, 9)] <- 1

# simulating a case with species very close to each other
# here species 2, 4, 3, 8  close to species 1
# so it is like commA and commB have only 2 species instead of 4 in the 2D space
traits.test0<-traits.test
traits.test0[c(2,5),]<-traits.test0[1,]+10^-6
traits.test0[c(3,8),]<-traits.test0[1,]+10^-6
traits.test0

## Not run:
# trying .core function with default qhull option
core.test0 <- functional.betapart.core(x = comm.test, traits = traits.test0, multi=FALSE,
                                       convhull.opt = list(conv1 = "Qt"))
core.test0 # crashing because of coplanarity
```

```
# trying new .core.pairwise with default qhull option for convex hull
core.pair_test0 <- functional.betapart.core.pairwise(x = comm.test, traits = traits.test0,
                                                     convhull.opt = list(conv1 = "Qt"))

# with default qhull options (Qt) it coud be impossible to compute the functional volumes
# this is why 'Qj' is now used as the default option for the convexhull volumes

# but other options could be passed to convexhull

# trying new core.pairwise with Qs for convex hull
core.pair_test0_Qs <- functional.betapart.core.pairwise(x = comm.test,
                                                        traits = traits.test0,
                                                       convhull.opt = list(conv1= "Qs"))
# not working

## End(Not run)

# trying new .pairwise with QJ (default option) for convex hull
core.pair_test0_Qj <- functional.betapart.core.pairwise(x = comm.test, traits = traits.test0,
                                                      convhull.opt = list(conv1 = "QJ"),
                                                        return.details = TRUE)
# OK and QJ applied to each volume computation
core.pair_test0_Qj

# equivalent to
core.pair_test0_Qj <- functional.betapart.core.pairwise(x = comm.test, traits = traits.test0,
                                                        return.details = TRUE)

# but QJ could be applied only when error are generated with other options :
core.pair_test0_Qja <- functional.betapart.core.pairwise(x = comm.test, traits = traits.test0,
                                                       convhull.opt = list(conv1 = "Qt",
                                                                           conv2 = "QJ"),
                                                         return.details = TRUE)
# OK and QJ applied only for one volume computation (community 'B')

core.pair_test0_Qja
# numerous intersection had to be computed with inter_rcdd

# the results are comparable
all.equal(core.pair_test0_Qj[-9], core.pair_test0_Qja[-9]) # -9 to remove details

# the pairwise functional functional betadiversity
functional.beta.pair(core.pair_test0_Qj, index.family = "jaccard")

## Not run:
##### using internal parallelisation to fasten pairiwse dissimilarity
# by default (parallel = FALSE) the code is run in serial
test.core.pair <- functional.betapart.core.pairwise(x = comm.test, traits = traits.test,
                                                    parallel = FALSE)
# when using internal parallelisation and default options
# it uses half of the cores and 1 task per run (this can be customised)
# test.core.pairp <- functional.betapart.core(x = comm.test, traits = traits.test,
```

```
#                                          multi = FALSE, return.details = FALSE,
#                                          fbc.step = FALSE, parallel = TRUE)
# you can set the number of core to use :
test.core.pairp <-
  functional.betapart.core.pairwise(x = comm.test, traits = traits.test,
                                    parallel = TRUE,
                                    opt.parallel = beta.para.control(nc = 2))
all.equal(test.core.pair, test.core.pairp)


# as inter_geom is much more faster than inter_rccd it is useful to increase the number
# of calculus run per iteration to limit the time consumed by the cluster itself
# you can play on size (this would not have sense here as there is only few computation)

  test.core.pairp <-
    functional.betapart.core.pairwise(x = comm.test, traits = traits.test,
                                      parallel = TRUE,
                                      opt.parallel =
                                      beta.para.control(nc = 2,
                                                        size = 100))
  all.equal(test.core.pair, test.core.pairp)


library(microbenchmark)
  microbenchmark(
    serial = functional.betapart.core.pairwise(comm.test, traits.test),
    nc2 = functional.betapart.core.pairwise(comm.test, traits.test,
                                            parallel = TRUE,
                                            opt.parallel = beta.para.control(nc = 2)),
    nc4 = functional.betapart.core.pairwise(comm.test, traits.test, multi = FALSE,
                                            parallel = TRUE,
                                            opt.parallel = beta.para.control(nc = 4))
  )


# If the number of species is very different among communities
# load-balancing parallelisation could be more efficient
# especially when the number of community is high
test.core.pairp <-
  functional.betapart.core.pairwise(comm.test, traits.test,
                                    parallel = TRUE,
                                    opt.parallel = beta.para.control(nc = 2, LB = TRUE))

# use you can use fork cluster (but not on Windows)
test.core.pairp <-
  functional.betapart.core.pairwise(comm.test, traits.test,
                                      parallel = TRUE,
                                      opt.parallel =
                                        beta.para.control(nc = 2, type = "FORK"))


# a progress bar can be displayed to asses the evolution of the computations
  test.core.pairp <-
```

```
    functional.betapart.core.pairwise(comm.test, traits.test,
                                      parallel = TRUE,
                                      opt.parallel =
                                        beta.para.control(nc = 2, LB = TRUE),
                                      progress = TRUE)


# using internal parallelisation is not always useful, especially on small data set
# load balancing is very helpful when species richness are highly variable

# Null model using 'external' parallel computing

# Example 1: pairwise functional beta diversity (functional.beta.pair)
# Note that this is an example with a small number of samples and null model
# permutations for illustration.
# Real null model analyses should have a much greater number of samples and permutations.

##### 4 communities in a 3D functional space

comm.test <- matrix(0, 4, 11, dimnames = list(c("A", "B", "C", "D"),
                                              paste("sp", 1:11, sep = "")))
comm.test["A", c(1, 2, 4, 5)] <- 1
comm.test["B", c(1, 3, 8, 9)] <- 1
comm.test["C", c(6, 7, 10, 11)] <- 1
comm.test["D", c( 2, 4, 7, 9)] <- 1

set.seed(1)
traits.test <- matrix(rnorm(11*3, mean = 0, sd = 1), 11, 3)
dimnames(traits.test) <- list(paste("sp", 1:11, sep = "") ,
                              c("Trait 1", "Trait 2", "Trait 3"))

# Required packages
library(doSNOW)
library(picante)
library(foreach)
library(itertools)

# define number of permutations for the null model (the usual is 1000)
# make sure that nperm/nc is a whole number so that all cores have the same number
# of permutations to work on
nperm <- 100

test.score <- functional.betapart.core.pairwise(comm.test, traits.test)

obs.pair.func.dis <- functional.beta.pair(x = test.score, index.family = "sorensen")

# transform functional.beta.pair results into a matrix
obs.pair.func.dis <- do.call(rbind, obs.pair.func.dis)

# set names for each pair of site
pair_names <- combn(rownames(comm.test), 2, FUN = paste, collapse = "_")
colnames(obs.pair.func.dis) <- pair_names
```

```
# define number of cores
# Use parallel::detectCores() to determine number of cores available in your machine
nc <- 2

# 4 cores would be better (nc <- 4)

# create cluster
cl <- snow::makeCluster(nc)

# register parallel backend
doSNOW:::registerDoSNOW(cl)

# export necessary variables and functions to the cluster of cores
snow::clusterExport(cl = cl, c("comm.test", "traits.test"),
                    envir = environment())

# creation of an iterator to run 1 comparaisons on each core at time
it <- itertools::isplitIndices(nperm, chunkSize = 10)

null.pair.func.dis <-
  foreach(n = it, .combine = c,
          .packages=c("picante","betapart","fastmatch", "rcdd", "geometry")) %dopar% {

            # it enables to adjust the number of permutations (nt) done on each run
            nt <- length(n)
            null.pair.temp <- vector(mode = "list", length = nt)

            # for each core "n" perform "nt" permutations
            for (j in 1:nt){

              # randomize community with chosen null model
              # for this particular example we used the "independent swap algorithm"
              # but the user can choose other types of permutation
              # or create it's own null model
             null.comm.test <- randomizeMatrix(comm.test, null.model = "independentswap",
                                               iterations=1000)

              # execute functional.betapart.core function
              null.test.score <-
                functional.betapart.core.pairwise(null.comm.test, traits = traits.test,
                                                  parallel = FALSE)
             # using 'external' parallelisation it is necessary to set parralel to FALSE

              # in this artificial example there are a few combinations of species that
             # the convex hull cannot be calculated due to some odd geometric combination
              # so we need to specify to use the 'QJ' options in case of errors

              # compute the pairwise beta-diversity null values and input them in the
              # temporary result matrix
             res <- functional.beta.pair(x = null.test.score, index.family = "sorensen")
              null.pair.temp[[j]] <- do.call(rbind, res)
            }
            #retrieve the results from each core
```

```
            null.pair.temp
          }

# stop cluster
snow::stopCluster(cl)

#compute the mean, standard deviation and p-values of dissimilarity metrics
# for each pair of site

mean.null.pair.func <- matrix(numeric(), ncol = ncol(obs.pair.func.dis),
                              nrow = nrow(obs.pair.func.dis))
sd.null.pair.func <- matrix(numeric(), ncol = ncol(obs.pair.func.dis),
                            nrow = nrow(obs.pair.func.dis))
p.pair.func.dis <- matrix(numeric(), ncol = ncol(obs.pair.func.dis),
                          nrow = nrow(obs.pair.func.dis))

# for each one of the 3 null dissimilarity metrics (SIN, SNE and SOR)
for (j in 1:nrow(obs.pair.func.dis)){
  matnull <- sapply(null.pair.func.dis, function(x) x[j,])
  mean.null.pair.func[j,] <- rowMeans(matnull)
  sd.null.pair.func[j,] <- sqrt(rowSums((matnull - mean.null.pair.func[j,])^2)/(nperm-1))
  p.pair.func.dis[j,] <- rowSums(matnull >= obs.pair.func.dis[j,])
 p.pair.func.dis[j,] <- (pmin(p.pair.func.dis[j,],nperm-p.pair.func.dis[j,])+1)/(nperm+1)
  # the +1 is to take into account that the observed value is one of the possibilities
}

# compute standardized effect sizes
ses.pair.func.dis <- (obs.pair.func.dis - mean.null.pair.func)/sd.null.pair.func

## End(Not run)
```

`inter_geom` *Internal function to compute convexhull volume*

Description

Estimation of the convexhull volume of the intersection of two hypervolumes based on the `intersectn` function

Usage

```
inter_geom(ps1, ps2, options = "Tv", tol = 0, fp = NULL, qhull.opt = "n FA")
```

Arguments

`ps1`	A matrix of coordinates.
`ps2`	A second matrix of coordinates.
`options`	Options pass to `halfspacen`.
`tol`	Tolerance, see `intersectn`.

fp	Coordinates of feasible point (NULL).
qhull.opt	qhull options.

Value

the convex hull volume of the intersection of two hypervolumes

See Also

inter_rcdd, intersectn

Examples

```
## Not run: mat1 <- matrix(runif(30), 10)
mat2 <- matrix(runif(30), 10)
inter_geom(mat1, mat2)
## End(Not run)
```

---

inter_geom_coord *Internal function to compute convexhull volume and vertice coordinates*

Description

Estimation of the convexhull volume and the vertices of the intersection of two hypervolumes based on geometry functions

Usage

```
inter_geom_coord(
  ps1,
  ps2,
  options = "Tv",
  tol = 0,
  fp = NULL,
  qhull.opt = "n FA"
)
```

Arguments

ps1	A matrix of coordinates.
ps2	A second matrix of coordinates.
options	Options pass to halfspacen.
tol	Tolerance, see intersectn.
fp	Coordinates of feasible point (NULL).
qhull.opt	qhull options.

Value

A list of 2 elements.

coord the vertice coordinates

vol a volume corresponding to the intersection of the two hypervolumes

See Also

inter_rcdd_coord

Examples

```
## Not run: mat1 <- matrix(runif(30), 10)
mat2 <- matrix(runif(30), 10)
inter_geom_coord(mat1, mat2)
## End(Not run)
```

---

inter_rcdd *Internal function to compute convexhull volume*

Description

Estimation of the convexhull volume of the intersection of two hypervolumes based on rcdd functions

Usage

```
inter_rcdd(set1, set2, qhull.opt = "FA", conv2 = function(...) NA)
```

Arguments

set1	A matrix of coordinates
set2	A matrix of coordinates
qhull.opt	Qhull options, see http://www.qhull.org/html/qh-optq.htm
conv2	A function applyed if the convexhull function crashes

Value

A volume corresponding to the intersection of the two hypervolumes

See Also

inter_geom

Examples

```
## Not run: mat1 <- matrix(runif(30), 10)
mat2 <- matrix(runif(30), 10)
inter_rcdd(mat1, mat2)
## End(Not run)
```

`inter_rcdd_coord` *Internal function to compute convexhull volume and vertice coordinates*

Description

Estimation of the convexhull volume and the vertices of the intersection of two hypervolumes based on rcdd functions

Usage

```
inter_rcdd_coord(set1, set2, qhull.opt = "FA", conv2 = function(...) NA)
```

Arguments

`set1`	A matrix of coordinates
`set2`	A matrix of coordinates
`qhull.opt`	Qhull options, see http://www.qhull.org/html/qh-optq.htm
`conv2`	A function applyed if the convexhull function crashes

Value

A list of 2 elements.

coord the vertice coordinates

vol a volume corresponding to the intersection of the two hypervolumes

See Also

`inter_geom_coord`

Examples

```
## Not run: mat1 <- matrix(runif(30), 10)
mat2 <- matrix(runif(30), 10)
inter_rcdd_coord(mat1, mat2)
## End(Not run)
```

phylo.beta.multi *Multiple-site phylogenetic dissimilarities*

Description

Computes 3 distance values accounting for the multiple-site phylogenetic turnover and nestedness components of phylogenetic beta diversity, and the sum of both values. Phylogenetic dissimilarities are based on Faith's phylogenetic diversity.

Usage

```
phylo.beta.multi(x, tree, index.family="sorensen")
```

Arguments

x	a community matrix or data frame, where rows are sites and columns are species. Alternatively x can be a phylo.betapart object derived from the phylo.betapart.core function
tree	a phylogenetic tree of class phylo with tips names identic to species names from the community matrix.
index.family	family of dissimilarity indices, partial match of "sorensen" or "jaccard".

Details

The Sorensen dissimilarity index allows computing the PhyloSor index (Bryant et al. 2008) whereas the Jaccard dissimilarity index allows computing the UniFrac index (Lozupone & Knight 2005).

Value

The function returns a list with three phylogenetic dissimilarity values.

For index.family="sorensen" the three values are:

phylo.beta.sim	dist object, dissimilarity value accounting for phylogenetic turnover, measured as Simpson derived multiple-site phylogenetic dissimilarity
phylo.beta.sne	dist object, dissimilarity value accounting for nestedness-resultant phylogenetic dissimilarity, measured as the nestedness-fraction of Sorensen derived multiple-site phylogenetic dissimilarity
phylo.beta.sor	dist object, dissimilarity value accounting for phylogenetic beta diversity, measured as Sorensen derived multiple-site phylogenetic dissimilarity

For index.family="jaccard" the three values are:

phylo.beta.jtu	dist object, dissimilarity value accounting for phylogenetic turnover, measured as the turnover-fraction of Jaccard derived multiple-site phylogenetic dissimilarity

`phylo.beta.jne` `dist` object, dissimilarity value accounting for nestedness-resultant phylogenetic dissimilarity, measured as the nestedness-fraction of Jaccard derived multiple-site phylogenetic dissimilarity

`phylo.beta.jac` `dist` object, dissimilarity value accounting for phylogenetic beta diversity, measured as Jaccard derived multiple-site phylogenetic dissimilarity

Author(s)

Julien De Bortoli (juldebortoli@yahoo.fr), Fabien Leprieur(fabien.leprieur@univ-montp2.fr), Andrés Baselga and David Orme

References

Baselga A. (2012) The relationship between species replacement, dissimilarity derived from nestedness, and nestedness. Global Ecology and Biogeography 21, 1223-1232

Bryant JA, Lamanna C, Morlon H, Kerkhoff AJ, Enquist BJ, et al. (2008) Microbes on mountainsides: Contrasting elevational patterns of bacterial and plant diversity. Proceedings of the National Academy of Sciences of the United States of America 105: 11505-11511.

Faith DP, Lozupone CA, Nipperess D, Knight R (2009) The Cladistic Basis for the Phylogenetic Diversity (PD) Measure Links Evolutionary Features to Environmental Gradients and Supports Broad Applications of Microbial Ecology's "Phylogenetic Beta Diversity" Framework. Int J Mol Sci 10: 4723-4741. doi: 10.3390/ijms10114723.

Leprieur F, Albouy C, De Bortoli J, Cowman PF, Bellwood DR, et al. (2012) Quantifying Phylogenetic Beta Diversity: Distinguishing between "True" Turnover of Lineages and Phylogenetic Diversity Gradients. PLoS ONE 7(8): e42760. doi:10.1371/journal.pone.0042760

Lozupone C, Knight R (2005) UniFrac: a new phylogenetic method for comparing microbial communities. Applied and Environmental Microbiology 71: 8228-8235.

See Also

`phylo.betapart.core`, `beta.multi`

Examples

```
# toy tree for 6 species (sp1 to sp6)
require(ape)
toy.tree<-read.tree(text="(((sp1:1,sp2:1):5,(sp3:3,sp4:3):3):2,(sp5:7,sp6:7):1);")
plot(toy.tree)

# toy community table with 6 assemblages (A to F) with 6 species (sp1 to sp6)
toy.comm<-matrix(nrow=6, ncol=6)
rownames(toy.comm)<-c("A","B","C","D","E","F")
colnames(toy.comm)<-c("sp1","sp2","sp3","sp4","sp5","sp6")
toy.comm[1,]<-c(1,1,1,0,0,0)
toy.comm[2,]<-c(0,1,1,1,0,0)
toy.comm[3,]<-c(0,0,1,1,1,0)
toy.comm[4,]<-c(0,0,1,1,1,1)
toy.comm[5,]<-c(0,0,0,1,1,1)
toy.comm[6,]<-c(1,0,0,1,1,1)
```

```
toy.phylobetamulti<-phylo.beta.multi(toy.comm, toy.tree, index.family="sor")
toy.betamulti<-beta.multi(toy.comm, index.family="sor")
```

`phylo.beta.pair` *Pair-wise phylogenetic dissimilarities*

Description

Computes 3 distance matrices accounting for the phylogenetic turnover and nestedness components of phylogenetic beta diversity, and the sum of both values. Phylogenetic dissimilarities are based on Faith's phylogenetic diversity.

Usage

```
phylo.beta.pair(x, tree, index.family="sorensen")
```

Arguments

`x`	a community matrix or data frame, where rows are sites and columns are species. Alternatively `x` can be a `phylo.betapart` object derived from the `phylo.betapart.core` function
`tree`	a phylogenetic tree of class phylo with tips names identic to species names from the community matrix.
`index.family`	family of dissimilarity indices, partial match of `"sorensen"` or `"jaccard"`.

Details

The Sorensen dissimilarity index allows computing the PhyloSor index (Bryant et al. 2008) whereas the Jaccard dissimilarity index allows computing the UniFrac index (Lozupone & Knight 2005).

Value

The function returns a list with three phylogenetic dissimilarity matrices.

For `index.family="sorensen"` the three matrices are:

`phylo.beta.sim`	`dist` object, dissimilarity matrix accounting for phylogenetic turnover, measured as Simpson derived pair-wise phylogenetic dissimilarity
`phylo.beta.sne`	`dist` object, dissimilarity matrix accounting for nestedness-resultant phylogenetic dissimilarity, measured as the nestedness-fraction of Sorensen derived pair-wise phylogenetic dissimilarity
`phylo.beta.sor`	`dist` object, dissimilarity matrix accounting for phylogenetic beta diversity, measured as Sorensen derived pair-wise phylogenetic dissimilarity

For `index.family="jaccard"` the three matrices are:

`phylo.beta.jtu` `dist` object, dissimilarity matrix accounting for phylogenetic turnover, measured as the turnover-fraction of Jaccard derived pair-wise phylogenetic dissimilarity

`phylo.beta.jne` `dist` object, dissimilarity matrix accounting for nestedness-resultant phylogenetic dissimilarity, measured as the nestedness-fraction of Jaccard derived pair-wise phylogenetic dissimilarity

`phylo.beta.jac` `dist` object, dissimilarity matrix accounting for phylogenetic beta diversity, measured as Jaccard derived pair-wise phylogenetic dissimilarity

Author(s)

Julien De Bortoli (juldebortoli@yahoo.fr), Fabien Leprieur(fabien.leprieur@univ-montp2.fr), Andrés Baselga and David Orme

References

Baselga A. (2012) The relationship between species replacement, dissimilarity derived from nestedness, and nestedness. Global Ecology and Biogeography 21, 1223-1232

Bryant JA, Lamanna C, Morlon H, Kerkhoff AJ, Enquist BJ, et al. (2008) Microbes on mountainsides: Contrasting elevational patterns of bacterial and plant diversity. Proceedings of the National Academy of Sciences of the United States of America 105: 11505-11511.

Faith DP, Lozupone CA, Nipperess D, Knight R (2009) The Cladistic Basis for the Phylogenetic Diversity (PD) Measure Links Evolutionary Features to Environmental Gradients and Supports Broad Applications of Microbial Ecology's "Phylogenetic Beta Diversity" Framework. Int J Mol Sci 10: 4723-4741. doi: 10.3390/ijms10114723.

Leprieur F, Albouy C, De Bortoli J, Cowman PF, Bellwood DR, et al. (2012) Quantifying Phylogenetic Beta Diversity: Distinguishing between "True" Turnover of Lineages and Phylogenetic Diversity Gradients. PLoS ONE 7(8): e42760. doi:10.1371/journal.pone.0042760

Lozupone C, Knight R (2005) UniFrac: a new phylogenetic method for comparing microbial communities. Applied and Environmental Microbiology 71: 8228-8235.

See Also

`phylo.betapart.core`, `beta.pair`

Examples

```
# toy tree for 6 species (sp1 to sp6)
require(ape)
toy.tree<-read.tree(text="(((sp1:1,sp2:1):5,(sp3:3,sp4:3):3):2,(sp5:7,sp6:7):1);")
plot(toy.tree)

# toy community table with 6 assemblages (A to F) with 6 species (sp1 to sp6)
toy.comm<-matrix(nrow=6, ncol=6)
rownames(toy.comm)<-c("A","B","C","D","E","F")
colnames(toy.comm)<-c("sp1","sp2","sp3","sp4","sp5","sp6")
toy.comm[1,]<-c(1,1,1,0,0,0)
toy.comm[2,]<-c(0,1,1,1,0,0)
```

```
toy.comm[3,]<-c(0,0,1,1,1,0)
toy.comm[4,]<-c(0,0,1,1,1,1)
toy.comm[5,]<-c(0,0,0,1,1,1)
toy.comm[6,]<-c(1,0,0,1,1,1)


toy.phylobetapair<-phylo.beta.pair(toy.comm, toy.tree, index.family="sor")
toy.betapair<-beta.pair(toy.comm, index.family="sor")
plot(toy.betapair$beta.sim,toy.phylobetapair$phylo.beta.sim)
plot(toy.betapair$beta.sne,toy.phylobetapair$phylo.beta.sne)
```

`phylo.betapart.core` *Core calculations of phylogenetic dissimilarities metrics*

Description

Computes the basic quantities needed for computing the multiple-site phylogenetic beta diversity measures and pairwise phylogenetic dissimilarity matrices.

Usage

```
phylo.betapart.core(x, tree)
```

Arguments

`x`	a community matrix or data frame, where rows are sites and columns are species.
`tree`	a phylogenetic tree of class phylo with tips names identic to species names from the community matrix.

Value

The function returns a list with:

`sumSi`	the sum of the phylogenetic diversity values of all sites
`St`	the total phylogenetic diversity in the dataset
`shared`	a matrix containing the phylogenetic diversity shared between pairs of sites
`sum.not.shared`	a matrix containing the total phylogenetic diversity not shared between pairs of sites: b+c
`max.not.shared`	a matrix containing the total maximum phylogenetic diversity not shared between pairs of sites: max(b,c)
`min.not.shared`	a matrix containing the total minimum phylogenetic diversity not shared between pairs of sites: min(b,c)

Author(s)

Julien De Bortoli (juldebortoli@yahoo.fr), Fabien Leprieur(fabien.leprieur@univ-montp2.fr), Andrés Baselga and David Orme

References

Baselga A. (2012) The relationship between species replacement, dissimilarity derived from nestedness, and nestedness. Global Ecology and Biogeography 21, 1223-1232

Bryant JA, Lamanna C, Morlon H, Kerkhoff AJ, Enquist BJ, et al. (2008) Microbes on mountainsides: Contrasting elevational patterns of bacterial and plant diversity. Proceedings of the National Academy of Sciences of the United States of America 105: 11505-11511.

Faith DP, Lozupone CA, Nipperess D, Knight R (2009) The Cladistic Basis for the Phylogenetic Diversity (PD) Measure Links Evolutionary Features to Environmental Gradients and Supports Broad Applications of Microbial Ecology's "Phylogenetic Beta Diversity" Framework. Int J Mol Sci 10: 4723-4741. doi: 10.3390/ijms10114723.

Leprieur F, Albouy C, De Bortoli J, Cowman PF, Bellwood DR, et al. (2012) Quantifying Phylogenetic Beta Diversity: Distinguishing between "True" Turnover of Lineages and Phylogenetic Diversity Gradients. PLoS ONE 7(8): e42760. doi:10.1371/journal.pone.0042760

Lozupone C, Knight R (2005) UniFrac: a new phylogenetic method for comparing microbial communities. Applied and Environmental Microbiology 71: 8228-8235.

See Also

phylo.beta.pair, phylo.beta.multi

Examples

```
# toy tree for 6 species (sp1 to sp6)
require(ape)
toy.tree<-read.tree(text="(((sp1:1,sp2:1):5,(sp3:3,sp4:3):3):2,(sp5:7,sp6:7):1);")
plot(toy.tree)

# toy community table with 6 assemblages (A to F) with 6 species (sp1 to sp6)
toy.comm<-matrix(nrow=6, ncol=6)
rownames(toy.comm)<-c("A","B","C","D","E","F")
colnames(toy.comm)<-c("sp1","sp2","sp3","sp4","sp5","sp6")
toy.comm[1,]<-c(1,1,1,0,0,0)
toy.comm[2,]<-c(0,1,1,1,0,0)
toy.comm[3,]<-c(0,0,1,1,1,0)
toy.comm[4,]<-c(0,0,1,1,1,1)
toy.comm[5,]<-c(0,0,0,1,1,1)
toy.comm[6,]<-c(1,0,0,1,1,1)

toy.phylocore<-phylo.betapart.core(toy.comm, toy.tree)
```

`plot.decay` *Plotting distance decay curves from models computed with decay.model()*

Description

Takes the output of decay.model() and plots a distance-decay curve, either a negative exponential or power law function as estimated with decay.model().

Usage

```
## S3 method for class 'decay'
plot(x, xlim=c(0,max(x$data.x)), ylim=c(0,1),
add=FALSE, remove.dots=FALSE, col="black", pch=1, lty=1, lwd=5, cex=1, ...)
```

Arguments

x	the output of decay.model().
xlim	the range of spatial distances to be plotted, default is from 0 to the maximum distance in the data.
ylim	the range of assemblage similarities or dissimilarities to be plotted, default is from 0 to 1.
add	add to the previous plot.
remove.dots	remove the dots from the plot, thus retaining just the decay curve.
col	colour used.
pch	symbol used for points.
lty	line type.
lwd	line width.
cex	scale of text and symbols.
...	other parameters for plotting functions.

Author(s)

Andrés Baselga

References

Gómez-Rodríguez, C. & Baselga, A. 2018. Variation among European beetle taxa in patterns of distance decay of similarity suggests a major role of dispersal processes. Ecography 41: 1825-1834

See Also

decay.model

Examples

```
# presence/absence tables for longhorn beetles of South and North Europe
data(ceram.s)
data(ceram.n)

# spatial coordinates of territories in South and North Europe
data(coords.s)
data(coords.n)

# dissimilarity matrices
ceram.s.sim<-beta.pair(ceram.s)$beta.sim
```

```
ceram.n.sim<-beta.pair(ceram.n)$beta.sim

# spatial distances in km
distgeo.s<-dist(coords.s[,1:2])
distgeo.n<-dist(coords.n[,1:2])

# Negative exponential models for the decay of similarity with spatial distance
decay.south<-decay.model(y=1-ceram.s.sim, x=distgeo.s, y.type="sim", model.type="exp")
decay.north<-decay.model(y=1-ceram.n.sim, x=distgeo.n, y.type="sim", model.type="exp")

# Plot the decay models
plot.decay(decay.south, col="red")
plot.decay(decay.north, col="blue", add=TRUE)

# Equivalent models for the increase of dissimilarity with spatial distance
increase.south<-decay.model(y=ceram.s.sim, x=distgeo.s, y.type="dissim", model.type="exp")
increase.north<-decay.model(y=ceram.n.sim, x=distgeo.n, y.type="dissim", model.type="exp")

# Plot the decay models
plot.decay(increase.south, col="red")
plot.decay(increase.north, col="blue", add=TRUE)
```

`qhull.opt` *Specifying control Values for convexhull volume estimation*

Description

Set the default value to use inside the `functional.betapart.core` and `functional.betapart.pairwise` functions. It defined the options to use with the `convhulln` fonction.

Usage

```
qhull.opt(conv1 = "QJ", conv2 = NULL)
```

Arguments

`conv1`	A character vector specifying qhull options.
`conv2`	A character vector specifying qhull options to use if the internal computation of convexhull volumes generates an error. By default (`NULL`) `NA` is returned.

Details

`conv1` defined options that will be use systematically when calling convhulln to estimate the convexhull volume of the intersection, while `conv2` whill be used only if the first call to convhulln would have generated an error. For the complete list of possible options see For the list of options see http://www.qhull.org/html/qh-optq.htm. By default, no option are passed which would generates `NA` if internal error. Setting one of the two elements to "QJ" can solve different issues with very close numerical estimation (difference lower than 1e-4 in our tests).

Value

A named list of two elements.

See Also

inter_rcdd, inter_geom, convhulln.

Examples

```
## Not run: qhull.opt()
qhull.opt(conv1 = 'QJ')
qhull.opt(conv1 = "Qt", conv2 = 'QJ')
## End(Not run)
```

---

zdep *Assessing the differences between parameters of two distance-decay models computed with decay.model().*

Description

Takes two distance-decay models fitted with decay.model() and assess via block-site bootstrap whether the parameters of both models are equal. Two tests are conducted independently, one for the equality of the first parameters of both models, another for the equality of the second parameters (slopes) of both models. The null hypothesis is that parameters of both models are equal.

Usage

```
zdep(m1, m2, resamples, st.val = c(1, 0))
```

Arguments

m1	first distance decay model, the output of decay.model().
m2	second distance decay model, the output of decay.model().
resamples	the number of bootstrap resamples.
st.val	starting values for the nonlinear model.

Value

The function returns a dataframe with the tests statistics (Z.dep) and respective p values for both model parameters.

Author(s)

Ramiro Martín-Devasa, Sara Martínez-Santalla, Carola Gómez-Rodríguez, Rosa M. Crujeiras, Andrés Baselga

References

Martín-Devasa R, Martínez-Santalla S, Gómez-Rodríguez C, Crujeiras RM, Baselga A. 2022. Comparing distance-decay parameters: a novel test under pairwise dependence. Ecological Informatics 72: 101894

Martínez-Santalla S, Martín-Devasa R, Gómez-Rodríguez C, Crujeiras RM, Baselga A. 2022. Assessing the non-linear decay of community similarity: permutation and site-block resampling significance tests. Journal of Biogeography 49: 968-978

See Also

decay.model, plot.decay, boot.coefs.decay

Examples

```
# presence/absence tables for longhorn beetles of South and North Europe
data(ceram.s)
data(ceram.n)

# spatial coordinates of territories in South and North Europe
data(coords.s)
data(coords.n)

# dissimilarity matrices
ceram.s.sim<-beta.pair(ceram.s)$beta.sim
ceram.n.sim<-beta.pair(ceram.n)$beta.sim

# spatial distances in km
distgeo.s<-dist(coords.s[,1:2])
distgeo.n<-dist(coords.n[,1:2])

# Negative exponential distance decay models
decay.south<-decay.model(y=1-ceram.s.sim, x=distgeo.s, y.type="sim", model.type="exp")
decay.north<-decay.model(y=1-ceram.n.sim, x=distgeo.n, y.type="sim", model.type="exp")

# Plot the decay models
plot.decay(decay.south, col="red")
plot.decay(decay.north, col="blue", add=TRUE)

# Assess North-South difference between intercepts and slopes
zdep(decay.south, decay.north, resamples=1000)
```

Index