Package 'FD'

October 12, 2022

Type Package

Title Measuring Functional Diversity (FD) from Multiple Traits, and Other Tools for Functional Ecology

Version 1.0-12.1

Date 2014-08-19

Author Etienne Laliberté, Pierre Legendre, Bill Shipley

Maintainer Etienne Laliberté <etiennelaliberte@gmail.com>

Description Computes different multidimensional FD indices. Implements a distance-based framework to measure FD that allows any number and type of functional traits, and can also consider species relative abundances. Also contains other useful tools for functional ecology.

License GPL-2

LazyLoad yes

LazyData yes

Depends ade4, ape, geometry, vegan

Encoding latin1

NeedsCompilation yes

Repository CRAN

Date/Publication 2022-05-02 13:01:39 UTC

R topics documented:

FD-package 2
dbFD 3
dummy 10
fdisp 10
functcomp 12
gowdis 14
mahaldis 16
maxent 18
maxent.test 21
simul.dbFD 24
tussock 26

Index **28**

FD-package *Measuring Functional Diversity from Multiple Traits, and Other Tools for Functional Ecology*

Description

FD is a package to compute different multidimensional functional diversity (FD) indices. It implements a distance-based framework to measure FD that allows any number and type of functional traits, and can also consider species relative abundances. It also contains other tools for functional ecologists (e.g. maxent).

Details

Package:	FD
Type:	Package
Version:	1.0-12
Date:	2014-08-19
License:	GPL-2
LazyLoad:	yes
LazyData:	yes

FD computes different multidimensional FD indices. To compute FD indices, a species-by-trait(s) matrix is required (or at least a species-by-species distance matrix). gowdis computes the Gower dissimilarity from different trait types (continuous, ordinal, nominal, or binary), and tolerates NAs. It can treat ordinal variables as described by Podani (1999), and can handle asymetric binary variables and variable weights. gowdis is called by dbFD, the main function of **FD**.

dbFD uses principal coordinates analysis (PCoA) to return PCoA axes, which are then used as 'traits' to compute FD. dbFD computes several multidimensional FD indices, including the three indices of Villéger et al. (2008): functional richness (FRic), functional evenness (FEve), and functional divergence (FDiv). It also computes functional dispersion (FDis) (Laliberté and Legendre 2010), Rao's quadratic entropy (Q) (Botta-Dukát 2005), a posteriori functional group richness (FGR), and the community-level weighted means of trait values (CWM), an index of functional composition. Some of these indices can be weighted by species abundances. dbFD includes several options for flexibility.

Author(s)

Etienne Laliberté, Pierre Legendre and Bill Shipley

Maintainer: Etienne Laliberté <etiennelaliberte@gmail.com> https://www.elaliberte.info/

References

Botta-Dukát, Z. (2005) Rao's quadratic entropy as a measure of functional diversity based on multiple traits. *Journal of Vegetation Science* **16**:533-540.

Laliberté, E. and P. Legendre (2010) A distance-based framework for measuring functional diversity from multiple traits. *Ecology* **91**:299-305.

Podani, J. (1999) Extending Gower's general coefficient of similarity to ordinal characters. *Taxon* **48**:331-340.

Villéger, S., N. W. H. Mason and D. Mouillot (2008) New multidimensional functional diversity indices for a multifaceted framework in functional ecology. *Ecology* **89**:2290-2301.

Examples

```
# examples with a dummy dataset

ex1 <- gowdis(dummy$trait)
ex1

ex2 <- functcomp(dummy$trait, dummy$abun)
ex2

ex3 <- dbFD(dummy$trait, dummy$abun)
ex3

# examples with real data from New Zealand short-tussock grasslands
# these examples may take a few seconds to a few minutes each to run

ex4 <- gowdis(tussock$trait)

ex5 <- functcomp(tussock$trait, tussock$abun)

# 'lingoes' correction used because 'sqrt' does not work in that case
ex6 <- dbFD(tussock$trait, tussock$abun, corr = "lingoes")

## Not run:
# ward clustering to compute FGR, cailliez correction
ex7 <- dbFD(tussock$trait, tussock$abun, corr = "cailliez",
calc.FGR = TRUE, clust.type = "ward")
# choose 'g' for number of groups
# 6 groups seems to make good ecological sense
ex7

# however, calinksi criterion in 'kmeans' suggests
# that 6 groups may not be optimal
ex8 <- dbFD(tussock$trait, tussock$abun, corr = "cailliez",
calc.FGR = TRUE, clust.type = "kmeans", km.sup.gr = 10)

## End(Not run)
```

Description

dbFD implements a flexible distance-based framework to compute multidimensional functional diversity (FD) indices. dbFD returns the three FD indices of Villéger et al. (2008): functional richness (FRic), functional evenness (FEve), and functional divergence (FDiv), as well functional dispersion (FDis; Laliberté and Legendre 2010), Rao's quadratic entropy (Q) (Botta-Dukát 2005), a posteriori functional group richness (FGR) (Petchey and Gaston 2006), and the community-level weighted means of trait values (CWM; e.g. Lavorel et al. 2008). Some of these FD indices consider species abundances. dbFD includes several options for flexibility.

Usage

```
dbFD(x, a, w, w.abun = TRUE, stand.x = TRUE,
    ord = c("podani", "metric"), asym.bin = NULL,
    corr = c("sqrt", "cailliez", "lingoes", "none"),
    calc.FRic = TRUE, m = "max", stand.FRic = FALSE,
    scale.RaoQ = FALSE, calc.FGR = FALSE, clust.type = "ward",
    km.inf.gr = 2, km.sup.gr = nrow(x) - 1, km.iter = 100,
    km.crit = c("calinski", "ssi"), calc.CWM = TRUE,
    CWM.type = c("dom", "all"), calc.FDiv = TRUE, dist.bin = 2,
    print.pco = FALSE, messages = TRUE)
```

Arguments

x	matrix or data frame of functional traits. Traits can be numeric, ordered, or factor. Binary traits should be numeric and only contain 0 and 1. character traits will be converted to factor. NAs are tolerated. x can also be a species-by-species distance matrix of class dist, in which case NAs are not allowed. When there is only one trait, x can be also be a numeric vector, an ordered factor, or a unordered factor. In all cases, species labels are required. .
a	matrix containing the abundances of the species in x (or presence-absence, i.e. 0 or 1). Rows are sites and species are columns. Can be missing, in which case dbFD assumes that there is only one community with equal abundances of all species. NAs will be replaced by 0. The number of species (columns) in a must match the number of species (rows) in x. In addition, the species labels in a and x must be identical and in the same order.
w	vector listing the weights for the traits in x. Can be missing, in which case all traits have equal weights.
w.abun	logical; should FDis, Rao's Q, FEve, FDiv, and CWM be weighted by the relative abundances of the species?
stand.x	logical; if all traits are numeric, should they be standardized to mean 0 and unit variance? If not all traits are numeric, Gower's (1971) standardization by the range is automatically used; see gowdis for more details.
ord	character string specifying the method to be used for ordinal traits (i.e. ordered). "podani" refers to Eqs. 2a-b of Podani (1999), while "metric" refers to his Eq. 3. Can be abbreviated. See gowdis for more details.

asym.bin — vector listing the asymmetric binary variables in x. See gowdis for more details.

corr — character string specifying the correction method to use when the species-by-species distance matrix cannot be represented in a Euclidean space. Options are "sqrt", "cailliez", "lingoes", or "none". Can be abbreviated. Default is "sqrt". See 'details' section.

calc.FRic — logical; should FRic be computed?

m — the number of PCoA axes to keep as 'traits' for calculating FRic (when FRic is measured as the convex hull volume) and FDiv. Options are: any integer > 1, "min" (maximum number of traits that allows the $s \geq 2^t$ condition to be met, where s is the number of species and t the number of traits), or "max" (maximum number of axes that allows the $s > t$ condition to be met). See 'details' section.

stand.FRic — logical; should FRic be standardized by the 'global' FRic that include all species, so that FRic is constrained between 0 and 1?

scale.RaoQ — logical; should Rao's Q be scaled by its maximal value over all frequency distributions? See divc.

calc.FGR — logical; should FGR be computed?

clust.type — character string specifying the clustering method to be used to create the dendrogram of species for FGR. Options are "ward", "single", "complete", "average", "mcquitty", "median", "centroid", and "kmeans". For "kmeans", other arguments also apply (km.inf.fr, km.sup.gr, km.iter, and km.crit). See hclust and cascadeKM for more details.

km.inf.gr — the number of groups for the partition with the smallest number of groups of the cascade (min). Only applies if calc.FGR is TRUE and clust.type is "kmeans". See cascadeKM for more details.

km.sup.gr — the number of groups for the partition with the largest number of groups of the cascade (max). Only applies if calc.FGR is TRUE and clust.type is "kmeans". See cascadeKM for more details.

km.iter — the number of random starting configurations for each value of K. Only applies if calc.FGR is TRUE and clust.type is "kmeans". See cascadeKM for more details.

km.crit — criterion used to select the best partition. The default value is "calinski" (Calinski-Harabasz 1974). The simple structure index "ssi" is also available. Only applies if calc.FGR is TRUE and clust.type is "kmeans". Can be abbreviated. See cascadeKM for more details.

calc.CWM — logical; should the community-level weighted means of trait values (CWM) be calculated? Can be abbreviated. See functcomp for more details.

CWM.type — character string indicating how nominal, binary and ordinal traits should be handled for CWM. See functcomp for more details.

calc.FDiv — logical; should FDiv be computed?

dist.bin — only applies when x is a single unordered factor, in which case x is coded using dummy variables. dist.bin is an integer between 1 and 10 specifying the appropriate distance measure for binary data. 2 (the default) refers to the simple matching coefficient (Sokal and Michener 1958). See dist.binary for the other options.

print.pco — logical; should the eigenvalues and PCoA axes be returned?

messages — logical; should warning messages be printed in the console?

Details

Typical usage is

```
dbFD(x, a, \dots)
```

If x is a matrix or a data frame that contains only continuous traits, no NAs, and that no weights are specified (i.e. w is missing), a species-species Euclidean distance matrix is computed via dist. Otherwise, a Gower dissimilarity matrix is computed via gowdis. If x is a distance matrix, it is taken as is.

When x is a single trait, species with NAs are first excluded to avoid NAs in the distance matrix. If x is a single continuous trait (i.e. of class numeric), a species-species Euclidean distance matrix is computed via dist. If x is a single ordinal trait (i.e. of class ordered), gowdis is used and argument ord applies. If x is a single nominal trait (i.e. an unordered factor), the trait is converted to dummy variables and a distance matrix is computed via dist.binary, following argument dist.bin.

Once the species-species distance matrix is obtained, dbFD checks whether it is Euclidean. This is done via is.euclid. PCoA axes corresponding to negative eigenvalues are imaginary axes that cannot be represented in a Euclidean space, but simply ignoring these axes would lead to biased estimations of FD. Hence in dbFD one of four correction methods are used, following argument corr. "sqrt" simply takes the square root of the distances. However, this approach does not always work for all coefficients, in which case dbFD will stop and tell the user to select another correction method. "cailliez" refers to the approach described by Cailliez (1983) and is implemented via cailliez. "lingoes" refers to the approach described by Lingoes (1971) and is implemented via lingoes. "none" creates a distance matrix with only the positive eigenvalues of the Euclidean representation via quasieuclid. See Legendre and Legendre (1998) and Legendre and Anderson (1999) for more details on these corrections.

Principal coordinates analysis (PCoA) is then performed (via dudi.pco) on the *corrected* species-species distance matrix. The resulting PCoA axes are used as the new 'traits' to compute the three indices of Villéger et al. (2008): FRic, FEve, and FDiv. For FEve, there is no limit on the number of traits that can be used, so all PCoA axes are used. On the other hand, FRic and FDiv both rely on finding the minimum convex hull that includes all species (Villéger et al. 2008). This requires more species than traits. To circumvent this problem, dbFD takes only a subset of the PCoA axes as traits via argument m. This, however, comes at a cost of loss of information. The quality of the resulting reduced-space representation is returned by qual.FRic, which is computed as described by Legendre and Legendre (1998) and can be interpreted as a R^2-like ratio.

In dbFD, FRic is generally measured as the convex hull volume, but when there is only one continuous trait it is measured as the range (or the range of the ranks for an ordinal trait). Conversely, when only nominal and ordinal traits are present, FRic is measured as the number of unique trait value combinations in a community. FEve and FDiv, but not FRic, can account for species relative abundances, as described by Villéger et al. (2008).

Functional dispersion (FDis; Laliberté and Legendre 2010) is computed from the *uncorrected* species-species distance matrix via fdisp. Axes with negatives eigenvalues are corrected following the approach of Anderson (2006). When all species have equal abundances (i.e. presence-absence data), FDis is simply the average distance to the centroid (i.e. multivariate dispersion) as originally described by Anderson (2006). Multivariate dispersion has been proposed as an index of beta diversity (Anderson et al. 2006). However, Laliberté and Legendre (2010) have extended it to a FD index. FDis can account for relative abundances by shifting the position of the centroid towards the most abundant species, and then computing a weighted average distance to this new centroid, using

again the relative abundances as weights (Laliberté and Legendre 2010). FDis has no upper limit and requires at least two species to be computed. For communities composed of only one species, dbFD returns a FDis value of 0. FDis is by construction unaffected by species richness, it can be computed from any distance or dissimilarity measure (Anderson et al. 2006), it can handle any number and type of traits (including more traits than species), and it is not strongly influenced by outliers.

Rao's quadratic entropy (Q) is computed from the *uncorrected* species-species distance matrix via divc. See Botta-Dukát (2005) for details. Rao's Q is conceptually similar to FDis, and simulations (via simul.dbFD) have shown high positive correlations between the two indices (Laliberté and Legendre 2010). Still, one potential advantage of FDis over Rao's Q is that in the unweighted case (i.e. with presence-absence data), it opens possibilities for formal statistical tests for differences in FD between two or more communities through a distance-based test for homogeneity of multivariate dispersions (Anderson 2006); see betadisper for more details.

Functional group richness (FGR) is based on the classification of the species by the user from visual inspection of a dengrogram. Method "kmeans" is also available by calling cascadeKM. In that case, the Calinski-Harabasz (1974) criterion or the simple structure index (SSI) can be used to estimate the number of functional groups; see cascadeKM for more details. FGR returns the number of functional groups per community, as well as the abundance of each group in each community.

The community-level means of trait values (CWM) is an index of functional composition (Lavorel et al. 2008), and is computed via functcomp. Species with NAs for a given trait are excluded for that trait.

Value

nbsp	vector listing the number of species in each community
sing.sp	vector listing the number of functionally singular species in each community. If all species are functionally different, sing.sp will be identical to nbsp.
FRic	vector listing the FRic of each community
qual.FRic	quality of the reduced-space representation required to compute FRic and FDiv.
FEve	vector listing the FEve of each community
FDiv	vector listing the FDiv of each community. Only returned if calc.FDiv is TRUE.
FDis	vector listing the FDis of each community
RaoQ	vector listing the Rao's quadratic entropy (Q) of each community
FGR	vector listing the FGR of each community. Only returned if calc.FGR is TRUE.
spfgr	vector specifying functional group membership for each species. Only returned if calc.FGR is TRUE.
gr.abun	matrix containing the abundances of each functional group in each community. Only returned if calc.FGR is TRUE.
CWM	data frame containing the community-level weighted trait means (CWM). Only returned if calc.CWM is TRUE.
x.values	eigenvalues from the PCoA. Only returned if print.pco is TRUE.
x.axes	PCoA axes. Only returned if print.pco is TRUE.

Warning

Users often report that dbFD crashed during their analysis. Generally this occurs under Windows, and is almost always due to the computation of convex hull volumes. Possible solutions are to choose calc.FRic = "FALSE", or to reduce the dimensionality of the trait matrix using the "m" argument.

Note

dbFD borrows code from the F_RED function of Villéger et al. (2008).

Author(s)

Etienne Laliberté <etiennelaliberte@gmail.com> https://www.elaliberte.info/

References

Anderson, M. J. (2006) Distance-based tests for homogeneity of multivariate dispersions. *Biometrics* **62**:245-253.

Anderson, M. J., K. E. Ellingsen and B. H. McArdle (2006) Multivariate dispersion as a measure of beta diversity. *Ecology Letters* **9**:683-693.

Botta-Dukát, Z. (2005) Rao's quadratic entropy as a measure of functional diversity based on multiple traits. *Journal of Vegetation Science* **16**:533-540.

Cailliez, F. (1983) The analytical solution of the additive constant problem. *Psychometrika* **48**:305-310.

Calinski, T. and J. Harabasz (1974) A dendrite method for cluster analysis. *Communications in Statistics* **3**:1-27.

Gower, J. C. (1971) A general coefficient of similarity and some of its properties. *Biometrics* **27**:857-871.

Laliberté, E. and P. Legendre (2010) A distance-based framework for measuring functional diversity from multiple traits. *Ecology* **91**:299-305.

Lavorel, S., K. Grigulis, S. McIntyre, N. S. G. Williams, D. Garden, J. Dorrough, S. Berman, F. Quétier, A. Thebault and A. Bonis (2008) Assessing functional diversity in the field - methodology matters! *Functional Ecology* **22**:134-147.

Legendre, P. and M. J. Anderson (1999) Distance-based redundancy analysis: testing multispecies responses in multifactorial ecological experiments. *Ecological Monographs* **69**:1-24.

Legendre, P. and L. Legendre (1998) *Numerical Ecology*. 2nd English edition. Amsterdam: Elsevier.

Lingoes, J. C. (1971) Some boundary conditions for a monotone analysis of symmetric matrices. *Psychometrika* **36**:195-203.

Podani, J. (1999) Extending Gower's general coefficient of similarity to ordinal characters. *Taxon* **48**:331-340.

Sokal, R. R. and C. D. Michener (1958) A statistical method for evaluating systematic relationships. *The University of Kansas Scientific Bulletin* **38**:1409-1438.

Villéger, S., N. W. H. Mason and D. Mouillot (2008) New multidimensional functional diversity indices for a multifaceted framework in functional ecology. *Ecology* **89**:2290-2301.

See Also

gowdis, functcomp, fdisp, simul.dbFD, divc, treedive, betadisper

Examples

```
# mixed trait types, NA's
ex1 <- dbFD(dummy$trait, dummy$abun)
ex1

# add variable weights
# 'cailliez' correction is used because 'sqrt' does not work
w<-c(1, 5, 3, 2, 5, 2, 6, 1)
ex2 <- dbFD(dummy$trait, dummy$abun, w, corr="cailliez")

# if 'x' is a distance matrix
trait.d <- gowdis(dummy$trait)
ex3 <- dbFD(trait.d, dummy$abun)
ex3

# one numeric trait, one NA
num1 <- dummy$trait[,1] ; names(num1) <- rownames(dummy$trait)
ex4 <- dbFD(num1, dummy$abun)
ex4

# one ordered trait, one NA
ord1 <- dummy$trait[,5] ; names(ord1) <- rownames(dummy$trait)
ex5 <- dbFD(ord1, dummy$abun)
ex5

# one nominal trait, one NA
fac1 <- dummy$trait[,3] ; names(fac1) <- rownames(dummy$trait)
ex6 <- dbFD(fac1, dummy$abun)
ex6

# example with real data from New Zealand short-tussock grasslands
# 'lingoes' correction used because 'sqrt' does not work in that case
ex7 <- dbFD(tussock$trait, tussock$abun, corr = "lingoes")


## Not run:
# calc.FGR = T, 'ward'
ex7 <- dbFD(dummy$trait, dummy$abun, calc.FGR = T)
ex7

# calc.FGR = T, 'kmeans'
ex8 <- dbFD(dummy$trait, dummy$abun, calc.FGR = T,
clust.type = "kmeans")
ex8

# ward clustering to compute FGR
ex9 <- dbFD(tussock$trait, tussock$abun,
corr = "cailliez", calc.FGR = TRUE, clust.type = "ward")
```

```
# choose 'g' for number of groups
# 6 groups seems to make good ecological sense
ex9

# however, calinksi criterion in 'kmeans' suggests
# that 6 groups may not be optimal
ex10 <- dbFD(tussock$trait, tussock$abun, corr = "cailliez",
calc.FGR = TRUE, clust.type = "kmeans", km.sup.gr = 10)

## End(Not run)
```

---

dummy *Dummy Functional Trait Dataset*

Description

A dummy dataset containing 8 species and 8 functional traits (2 continuous, 2 nominal, 2 ordinal, and 2 binary), with some missing values. Also includes a matrix of species abundances from 10 communities.

Usage

```
dummy
```

Format

trait data frame of 8 functional traits on 8 species

abun matrix of abundances of the 8 species from 10 communities

Source

Etienne Laliberté <etiennelaliberte@gmail.com>

---

fdisp *Functional Dispersion*

Description

`fdisp` measures the functional dispersion (FDis) of a set of communities, as described by Laliberté and Legendre (2010).

Usage

```
fdisp(d, a, tol = 1e-07)
```

Arguments

d	a species-by- species distance matrix computed from functional traits, such as that returned by dist or gowdis. NAs are not allowed.
a	matrix containing the abundances of the species in d (or presence-absence, i.e. 0 or 1). Rows are sites and species are columns. Can be missing, in which case fdisp assumes that there is only one community with equal abundances of all species. NAs will be replaced by 0. The number of species (columns) in a must match the number of species in d. In addition, the species labels in a and d must be identical and in the same order.
tol	tolerance threshold to test whether the distance matrix is Euclidean : an eigenvalue is considered positive if it is larger than -tol*λ_1, where λ_1 is the largest eigenvalue.

Details

fdisp computes, for a set of communities, the average distance of individual objects (species) in PCoA space from any distance or dissimilarity measure, as described by Anderson (2006). The average distance to the centroid is a measure of multivariate dispersion and as been suggested as an index of beta diversity (Anderson et al. 2006). However, in fdisp both the centroid and the average distance to this centroid can be weighted by individual objects. In other words, fdisp returns the weighted average distance to the weighted centroid. This was suggested so that multivariate dispersion could be used as a multidimensional functional diversity (FD) index that can be weighted by species abundances. This FD index has been called functional dispersion (FDis) and is described by Laliberté and Legendre (2010).

In sum, FDis can account for relative abundances by shifting the position of the centroid towards the most abundant species, and then computing a weighted average distance to this new centroid, using again the relative abundances as weights (Laliberté and Legendre 2010). FDis has no upper limit and requires at least two species to be computed. For communities composed of only one species, dbFD returns a FDis value of 0. FDis is by construction unaffected by species richness, it can be computed from any distance or dissimilarity measure (Anderson et al. 2006), it can handle any number and type of traits (including more traits than species), and it is not strongly influenced by outliers.

FDis is conceptually similar to Rao's quadratic entropy Q (Botta-Dukát 2005), and simulations (via simul.dbFD) have shown high positive correlations between the two indices (Laliberté and Legendre 2010). Still, one potential advantage of FDis over Rao's Q is that in the unweighted case (i.e. with presence-absence data), it opens possibilities for formal statistical tests for differences in FD between two or more communities through a distance-based test for homogeneity of multivariate dispersions (Anderson 2006); see betadisper for more details.

Corrections for PCoA axes corresponding to negative eigenvalues are applied following Anderson (2006); see also betadisper for more details on these corrections.

Value

FDis	vector listing the FDis of each community
eig	vector listing the eigenvalues of the PCoA
vectors	matrix containing the PCoA axes

Note

fdisp is implemented in dbFD and is used to compute the functional dispersion (FDis) index.

Author(s)

Etienne Laliberté <etiennelaliberte@gmail.com> https://www.elaliberte.info/

References

Anderson, M. J. (2006) Distance-based tests for homogeneity of multivariate dispersions. *Biometrics* **62**:245-253.

Anderson, M. J., K. E. Ellingsen and B. H. McArdle (2006) Multivariate dispersion as a measure of beta diversity. *Ecology Letters* **9**:683-693.

Botta-Dukát, Z. (2005) Rao's quadratic entropy as a measure of functional diversity based on multiple traits. *Journal of Vegetation Science* **16**:533-540.

Laliberté, E. and P. Legendre (2010) A distance-based framework for measuring functional diversity from multiple traits. *Ecology* **91**299:305.

See Also

dbFD for computing multidimensional FD indices and betadisper from which fdisp borrows some code.

Examples

```
# dummy dataset
dummy.dist <- gowdis(dummy$trait)
ex1 <- fdisp(dummy.dist, dummy$abun)
ex1

# example with real data from New Zealand short-tussock grasslands
ex2 <- fdisp(gowdis(tussock$trait), tussock$abun)
ex2
```

---

functcomp *Functional Composition*

Description

functcomp returns the functional composition of a set of communities, as measured by the community-level weighted means of trait values (CWM; e.g. Lavorel et al. 2008).

Usage

```
functcomp(x, a, CWM.type = c("dom", "all"), bin.num = NULL)
```

Arguments

x	matrix or data frame containing the functional traits. Traits can be numeric, ordered, or factor. Binary traits should be numeric and only contain 0 and 1. character traits will be converted to factor. For a given trait, species with NA are excluded.
a	matrix containing the abundances of the species in x (or presence-absence, i.e. 0 or 1). Rows are sites and species are columns. The number of species (columns) in a must match the number of species (rows) in x. In addition, the species labels in a and x must be identical and in the same order. NAs will be replaced by 0.
CWM.type	character string indicating how nominal, binary and ordinal traits should be handled. See 'details'.
bin.num	vector indicating binary traits to be treated as continuous.

Details

functcomp computes the community-level weighted means of trait values for a set of communities (i.e. sites). For a continuous trait, CWM is the mean trait value of all species present in the community (after excluding species with NAs), weighted by their relative abundances.

For ordinal, nominal and binary traits, either the dominant class is returned (when CWM.type is "dom"), or the abundance of each individual class is returned (when CWM.type is "all").

The default behaviour of binary traits being treated as nominal traits can be over-ridden by specifying bin.num, in which case they are treated as numeric traits.

When CWM.type = "dom", if the maximum abundance value is shared between two or more classes, then one of these classes is randomly selected for CWM. Because species with NAs for a given trait are excluded for that trait, it is possible that when CWM.type is set to "all", the sum of the abundances of all classes for a given ordinal/nominal/binary trait does not equal the sum of the species abundances. Thus, it is definitely not recommended to have NAs for very abundant species, as this will lead to biased estimates of functional composition.

Value

a data frame containing the CWM values of each trait for each community.

Note

functcomp is implemented in dbFD and will be returned if calc.CWM is TRUE.

Author(s)

Etienne Laliberté <etiennelaliberte@gmail.com> https://www.elaliberte.info/

References

Lavorel, S., K. Grigulis, S. McIntyre, N. S. G. Williams, D. Garden, J. Dorrough, S. Berman, F. Quétier, A. Thébault and A. Bonis (2008) Assessing functional diversity in the field - methodology matters! *Functional Ecology* **22**:134-147.

See Also

dbFD for measuring distance-based multidimensional functional diversity indices, including CWM.

Examples

```
# for ordinal, nominal and binary variables
# returns only the most frequent class
ex1 <- functcomp(dummy$trait, dummy$abun)
ex1

# returns the frequencies of each class
ex2 <- functcomp(dummy$trait, dummy$abun, CWM.type = "all")
ex2

# example with real data from New Zealand short-tussock grasslands
ex3 <- functcomp(tussock$trait, tussock$abun)
ex3
```

---

gowdis *Gower Dissimilarity*

Description

gowdis measures the Gower (1971) dissimilarity for mixed variables, including asymmetric binary variables. Variable weights can be specified. gowdis implements Podani's (1999) extension to ordinal variables.

Usage

```
gowdis(x, w, asym.bin = NULL, ord = c("podani", "metric", "classic"))
```

Arguments

x	matrix or data frame containing the variables. Variables can be numeric, ordered, or factor. Symmetric or asymmetric binary variables should be numeric and only contain 0 and 1. character variables will be converted to factor. NAs are tolerated.
w	vector listing the weights for the variables in x. Can be missing, in which case all variables have equal weights.
asym.bin	vector listing the asymmetric binary variables in x.
ord	character string specifying the method to be used for ordinal variables (i.e. ordered). "podani" refers to Eqs. 2a-b of Podani (1999), while "metric" refers to his Eq. 3 (see 'details'); both options convert ordinal variables to ranks. "classic" simply treats ordinal variables as continuous variables. Can be abbreviated.

Details

gowdis computes the Gower (1971) similarity coefficient exactly as described by Podani (1999), then converts it to a dissimilarity coefficient by using $D = 1 - S$. It integrates variable weights as described by Legendre and Legendre (1998).

Let $\mathbf{X} = \{x_{ij}\}$ be a matrix containing n objects (rows) and m columns (variables). The similarity G_{jk} between objects j and k is computed as

$$G_{jk} = \frac{\sum_{i=1}^{n} w_{ijk} s_{ijk}}{\sum_{i=1}^{n} w_{ijk}}$$

,

where w_{ijk} is the weight of variable i for the j-k pair, and s_{ijk} is the partial similarity of variable i for the j-k pair,

and where $w_{ijk} = 0$ if objects j and k cannot be compared because x_{ij} or x_{ik} is unknown (i.e. NA).

For binary variables, $s_{ijk} = 0$ if $x_{ij} \neq x_{ik}$, and $s_{ijk} = 1$ if $x_{ij} = x_{ik} = 1$ or if $x_{ij} = x_{ik} = 0$.

For asymmetric binary variables, same as above except that $w_{ijk} = 0$ if $x_{ij} = x_{ik} = 0$.

For nominal variables, $s_{ijk} = 0$ if $x_{ij} \neq x_{ik}$ and $s_{ijk} = 1$ if $x_{ij} = x_{ik}$.

For continuous variables,

$$s_{ijk} = 1 - \frac{|x_{ij} - x_{ik}|}{x_{i.max} - x_{i.min}}$$

where $x_{i.max}$ and $x_{i.min}$ are the maximum and minimum values of variable i, respectively.

For ordinal variables, when ord = "podani" or ord = "metric", all x_{ij} are replaced by their ranks r_{ij} determined over all objects (such that ties are also considered), and then

if ord = "podani"

$s_{ijk} = 1$ if $r_{ij} = r_{ik}$, otherwise

$$s_{ijk} = 1 - \frac{|r_{ij} - r_{ik}| - (T_{ij} - 1)/2 - (T_{ik} - 1)/2}{r_{i.max} - r_{i.min} - (T_{i.max} - 1)/2 - (T_{i.min} - 1)/2}$$

where T_{ij} is the number of objects which have the same rank score for variable i as object j (including j itself), T_{ik} is the number of objects which have the same rank score for variable i as object k (including k itself), $r_{i.max}$ and $r_{i.min}$ are the maximum and minimum ranks for variable i, respectively, $T_{i,max}$ is the number of objects with the maximum rank, and $T_{i.min}$ is the number of objects with the minimum rank.

if ord = "metric"

$$s_{ijk} = 1 - \frac{|r_{ij} - r_{ik}|}{r_{i.max} - r_{i.min}}$$

When ord = "classic", ordinal variables are simply treated as continuous variables.

Value

an object of class dist with the following attributes: Labels, Types (the variable types, where 'C' is continuous/numeric, 'O' is ordinal, 'B' is symmetric binary, 'A' is asymmetric binary, and 'N' is nominal), Size, Metric.

Author(s)

Etienne Laliberté <etiennelaliberte@gmail.com> https://www.elaliberte.info/, with some help from Philippe Casgrain for the C interface.

References

Gower, J. C. (1971) A general coefficient of similarity and some of its properties. *Biometrics* **27**:857-871.

Legendre, P. and L. Legendre (1998) *Numerical Ecology*. 2nd English edition. Amsterdam: Elsevier.

Podani, J. (1999) Extending Gower's general coefficient of similarity to ordinal characters. *Taxon* **48**:331-340.

See Also

daisy is similar but less flexible, since it does not include variable weights and does not treat ordinal variables as described by Podani (1999). Using ord = "classic" reproduces the behaviour of daisy.

Examples

```
ex1 <- gowdis(dummy$trait)
ex1

# check attributes
attributes(ex1)

# to include weights
w <- c(4,3,5,1,2,8,3,6)
ex2 <- gowdis(dummy$trait, w)
ex2

# variable 7 as asymmetric binary
ex3 <- gowdis(dummy$trait, asym.bin = 7)
ex3

# example with trait data from New Zealand vascular plant species
ex4 <- gowdis(tussock$trait)
```

---

mahaldis *Mahalanobis Distance*

Description

mahaldis measures the pairwise Mahalanobis (1936) distances between individual objects.

Usage

```
mahaldis(x)
```

Arguments

x	matrix containing the variables. NAs are not tolerated.

Details

mahaldis computes the Mahalanobis (1936) distances between individual objects. The Mahalanobis distance takes into account correlations among variables and does not depend on the scales of the variables.

mahaldis builds on the fact that type-II principal component analysis (PCA) preserves the Mahalanobis distance among objects (Legendre and Legendre 2012). Therefore, mahaldis first performs a type-II PCA on standardized variables, and then computes the Euclidean distances among (repositioned) objects whose positions are given in the matrix $\mathbf{G}$. This is equivalent to the Mahalanobis distances in the space of the original variables (Legendre and Legendre 2012).

Value

an object of class dist.

Author(s)

Pierre Legendre <pierre.legendre@umontreal.ca>

http://adn.biol.umontreal.ca/~numericalecology/

Ported to **FD** by Etienne Laliberté.

References

Legendre, P. and L. Legendre (2012) *Numerical Ecology*. 3nd English edition. Amsterdam: Elsevier.

See Also

mahalanobis computes the Mahalanobis distances among groups of objects, not individual objects.

Examples

```
mat <- matrix(rnorm(100), 50, 20)

ex1 <- mahaldis(mat)

# check attributes
attributes(ex1)
```

`maxent` *Estimating Probabilities via Maximum Entropy: Improved Iterative Scaling*

Description

`maxent` returns the probabilities that maximize the entropy conditional on a series of constraints that are linear in the features. It relies on the Improved Iterative Scaling algorithm of Della Pietra et al. (1997). It has been used to predict the relative abundances of a set of species given the trait values of each species and the community-aggregated trait values at a site (Shipley et al. 2006; Shipley 2009; Sonnier et al. 2009).

Usage

```
maxent(constr, states, prior, tol = 1e-07, lambda = FALSE)
```

Arguments

`constr`	vector of macroscopical constraints (e.g. community-aggregated trait values). Can also be a matrix or data frame, with constraints as columns and data sets (e.g. sites) as rows.
`states`	vector, matrix or data frame of states (columns) and their attributes (rows).
`prior`	vector, matrix or data frame of prior probabilities of states (columns). Can be missing, in which case a maximally uninformative prior is assumed (i.e. uniform distribution).
`tol`	tolerance threshold to determine convergence. See 'details' section.
`lambda`	Logical. Should λ-values be returned?

Details

The biological model of community assembly through trait-based habitat filtering (Keddy 1992) has been translated mathematically via a maximum entropy (maxent) model by Shipley et al. (2006) and Shipley (2009). A maxent model contains three components: (i) a set of possible states and their attributes, (ii) a set of macroscopic empirical constraints, and (iii) a prior probability distribution $\mathbf{q} = [q_j]$.

In the context of community assembly, states are species, macroscopic empirical constraints are community-aggregated traits, and prior probabilities $\mathbf{q}$ are the relative abundances of species of the regional pool (Shipley et al. 2006, Shipley 2009). By default, these prior probabilities $\mathbf{q}$ are maximally uninformative (i.e. a uniform distribution), but can be specificied otherwise (Shipley 2009, Sonnier et al. 2009).

To facilitate the link between the biological model and the mathematical model, in the following description of the algorithm states are species and constraints are traits.

Note that if `constr` is a matrix or data frame containing several sets (rows), a maxent model is run on each individual set. In this case if `prior` is a vector, the same prior is used for each set. A

different prior can also be specified for each set. In this case, the number of rows in `prior` must be equal to the number of rows in `constr`.

If $\mathbf{q}$ is not specified, set $p_j = 1/S$ for each of the S species (i.e. a uniform distribution), where p_j is the probability of species j, otherwise $p_j = q_j$.

Calulate a vector $\mathbf{c} = [c_i] = \{c_1,\ c_2,\ \ldots,\ c_T\}$, where $c_i = \sum_{j=1}^{S} t_{ij}$; i.e. each c_i is the sum of the values of trait i over all species, and T is the number of traits.

Repeat for each iteration k until convergence:

1. For each trait t_i (i.e. row of the constraint matrix) calculate:

$$\gamma_i(k) = ln\left(\frac{\bar{t}_i}{\sum_{j=1}^{S}\left(p_j(k)\ t_{ij}\right)}\right)\left(\frac{1}{c_i}\right)$$

This is simply the natural log of the known community-aggregated trait value to the calculated community-aggregated trait value at this step in the iteration, given the current values of the probabilities. The whole thing is divided by the sum of the known values of the trait over all species.

2. Calculate the normalization term Z:

$$Z(k) = \left(\sum_{j=1}^{S} p_j(k)\ e^{\left(\sum_{i=1}^{T}\gamma_i(k)\ t_{ij}\right)}\right)$$

3. Calculate the new probabilities p_j of each species at iteration $k+1$:

$$p_j(k+1) = \frac{p_j(k)\ e^{\left(\sum_{i=1}^{T}\gamma_i(k)\ t_{ij}\right)}}{Z(k)}$$

4. If $|max\left(p\left(k+1\right) - p\left(k\right)\right)| \leq$ tolerance threshold (i.e. argument `tol`) then stop, else repeat steps 1 to 3.

When convergence is achieved then the resulting probabilities ($\hat{p}_j$) are those that are as close as possible to q_j while simultaneously maximize the entropy conditional on the community-aggregated traits. The solution to this problem is the Gibbs distribution:

$$\hat{p}_j = \frac{q_j e^{\left(-\sum_{i=1}^{T}\lambda_i t_{ij}\right)}}{\sum_{j=1}^{S} q_j e^{\left(-\sum_{i=1}^{T}\lambda_i t_{ij}\right)}} = \frac{q_j e^{\left(-\sum_{i=1}^{T}\lambda_i t_{ij}\right)}}{Z}$$

This means that one can solve for the Langrange multipliers (i.e. weights on the traits, λ_i) by solving the linear system of equations:

$$\begin{pmatrix} ln(\hat{p}_1) \\ ln(\hat{p}_2) \\ \vdots \\ ln(\hat{p}_S) \end{pmatrix} = (\lambda_1,\ \lambda_2,\ \ldots,\ \lambda_T) \begin{bmatrix} t_{11} & t_{12} & \ldots & t_{1S} - ln(Z) \\ t_{21} & t_{22} & \vdots & t_{2S} - ln(Z) \\ \vdots & \vdots & \vdots & \vdots \\ t_{T1} & t_{T2} & \ldots & t_{TS} - ln(Z) \end{bmatrix} - ln(Z)$$

This system of linear equations has $T+1$ unknowns (the T values of λ plus $ln(Z)$) and S equations. So long as the number of traits is less than $S-1$, this system is soluble. In fact, the solution is the well-known least squares regression: simply regress the values $ln(\hat{p}_j)$ of each species on the trait values of each species in a multiple regression.

The intercept is the value of $ln(Z)$ and the slopes are the values of λ_i and these slopes (Lagrange multipliers) measure by how much the $ln(\hat{p}_j)$, i.e. the ln(relative abundances), changes as the value of the trait changes.

`maxent.test` provides permutation tests for maxent models (Shipley 2010).

Value

`prob`	vector of predicted probabilities
`moments`	vector of final moments
`entropy`	Shannon entropy of `prob`
`iter`	number of iterations required to reach convergence
`lambda`	λ-values, only returned if `lambda = T`
`constr`	macroscopical constraints
`states`	states and their attributes
`prior`	prior probabilities

Author(s)

Bill Shipley <bill.shipley@usherbrooke.ca>

http://www.billshipley.recherche.usherbrooke.ca/

Ported to **FD** by Etienne Laliberté.

References

Della Pietra, S., V. Della Pietra, and J. Lafferty (1997) Inducing features of random fields. *IEEE Transactions Pattern Analysis and Machine Intelligence* **19**:1-13.

Keddy, P. A. (1992) Assembly and response rules: two goals for predictive community ecology. *Journal of Vegetation Science* **3**:157-164.

Shipley, B., D. Vile, and É. Garnier (2006) From plant traits to plant communities: a statistical mechanistic approach to biodiversity. *Science* **314**: 812–814.

Shipley, B. (2009) From Plant Traits to Vegetation Structure: Chance and Selection in the Assembly of Ecological Communities. Cambridge University Press, Cambridge, UK. 290 pages.

Shipley, B. (2010) Inferential permutation tests for maximum entropy models in ecology. *Ecology* **in press**.

Sonnier, G., Shipley, B., and M. L. Navas. 2009. Plant traits, species pools and the prediction of relative abundance in plant communities: a maximum entropy approach. *Journal of Vegetation Science* **in press**.

See Also

`functcomp` to compute community-aggregated traits, and `maxent.test` for the permutation tests proposed by Shipley (2010).

Another faster version of `maxent` for multicore processors called `maxentMC` is available from Etienne Laliberté (<etiennelaliberte@gmail.com>). It's exactly the same as `maxent` but makes use of the **multicore**, **doMC**, and **foreach** packages. Because of this, `maxentMC` only works on POSIX-compliant OS's (essentially anything but Windows).

Examples

```
# an unbiased 6-sided dice, with mean = 3.5
# what is the probability associated with each side,
# given this constraint?
maxent(3.5, 1:6)

# a biased 6-sided dice, with mean = 4
maxent(4, 1:6)

# example with tussock dataset
traits <- tussock$trait[, c(2:7, 11)] # use only continuous traits
traits <- na.omit(traits) # remove 2 species with NA's
abun <- tussock$abun[, rownames(traits)] # abundance matrix
abun <- t(apply(abun, 1, function(x) x / sum(x) )) # relative abundances
agg <- functcomp(traits, abun) # community-aggregated traits
traits <- t(traits) # transpose matrix

# run maxent on site 1 (first row of abun), all species
pred.abun <- maxent(agg[1,], traits)

## Not run:
# do the constraints give predictive ability?
maxent.test(pred.abun, obs = abun[1,], nperm = 49)

## End(Not run)
```

`maxent.test` *Inferential Permutation Tests for Maximum Entropy Models*

Description

`maxent.test` performs the permutation tests proposed by Shipley (2010) for maximum entropy models. Two different null hypotheses can be tested: 1) the information encoded in the *entire set* of constraints $\mathbf{C}$ is irrelevant for predicting the probabilities, and 2) the information encoded in *subset* $\mathbf{B}$ of the entire set of constraints $\mathbf{C} = \mathbf{A} \cup \mathbf{B}$ is irrelevant for predicting the probabilities. A plot can be returned to facilitate interpretation.

Usage

```
maxent.test(model, obs, sub.c, nperm = 99, quick = TRUE,
alpha = 0.05, plot = TRUE)
```

Arguments

`model`	list returned by `maxent`.
`obs`	vector, matrix or data frame of observed probabilities of the states (columns).
`sub.c`	character or numeric vector specifying the subset of constraints $\mathbf{B}$ associated with null hypothesis 2. If missing, null hypothesis 1 is tested.
`nperm`	number of permutations for the test.
`quick`	if `TRUE`, the algorithm stops when `alpha` is outside the confidence interval of the *P*-value. Can be useful to speed up the routine.
`alpha`	desired alpha-level for the test. Only relevant if `quick` is `TRUE`.
`plot`	if `TRUE`, a plot is returned to facilitate interpretation.

Details

`maxent.test` is a direct translation of the permutation tests described by Shipley (2010). Please refer to this article for details.

Using `quick = FALSE` will return the true null probability for a given `nperm`. However, if `nperm` is large (a rule-of-thumb is $>= 999$ permutations for allowing inference at $\alpha = 0.05$), this can take a very long time. Using `quick = TRUE` is a much faster and highly recommended alternative if one is only interested in accepting/rejecting the null hypothesis at the specified α-level given by argument `alpha`.

If `maxent` was run with multiple data sets (i.e. if `constr` had more than one row), then `maxent.test` performs the test for all sets simultaneously, following the 'omnibus' procedure described by Shipley (2010).

The following measure of fit between observed and predicted probabilities is returned:

$$\text{fit} = 1 - \frac{\sum_{j=1}^{D}\sum_{i=1}^{S}(o_{ij} - p_{ij})^2}{\sum_{j=1}^{D}\sum_{i=1}^{S}(o_{ij} - q_{ij})^2}$$

where o_{ij}, p_{ij}, and q_{ij} are the observed, predicted and prior probabilities of state i from data set j, respectively, S is the number of states, and D the number of data sets (i.e. rows in `obs`). A value of

1 indicates perfect predictive capacity, while a value near zero indicates that the constraints provide no additional information beyond what is already contained in the prior $\mathbf{q}$ (Sonnier et al. 2009).

Value

`fit`	measure of fit giving the predictive ability of the entire set of constraints $\mathbf{C}$, beyond that already provided by the prior distribution.
`fit.a`	measure of fit giving the predictive ability of the subset of constraints $\mathbf{A}$, beyond that already provided by the prior distribution; only returned if `sub.c` is specified
`r2`	Pearson r^2 between observed and predicted probabilities, using the entire set of constraints $\mathbf{C}$
`r2.a`	Pearson r^2 between observed and predicted probabilities, using the subset of constraints $\mathbf{A}$; only returned if `sub.c` is specified
`r2.q`	Pearson r^2 between observed and prior probabilities; only returned when `sub.c` is missing
`obs.stat`	observed statistic used for the permutation test; see Shipley (2010)
`nperm`	number of permutations; can be smaller than the specified `nperm` when `quick` is `TRUE`
`pval`	*P*-value
`ci.pval`	approximate confidence intervals of the *P*-value

Warning

`maxent.test` is a computationally intensive function. The tests can take a very long time when `nperm` is large and `quick = FALSE`. It is highly recommended to use `quick = TRUE` because of this, unless you are interested in obtaining the true null probability.

Author(s)

Etienne Laliberté <etiennelaliberte@gmail.com>

https://www.elaliberte.info/

References

Sonnier, G., Shipley, B., and M. L. Navas. 2009. Plant traits, species pools and the prediction of relative abundance in plant communities: a maximum entropy approach. *Journal of Vegetation Science* **in press**.

Shipley, B. (2010) Inferential permutation tests for maximum entropy models in ecology. *Ecology* **in press**.

See Also

`maxent` to run the maximum entropy model that is required by `maxent.test`.

Another faster version of `maxent.test` for multicore processors called `maxent.testMC` is available from Etienne Laliberté (<etiennelaliberte@gmail.com>). It's exactly the same as `maxent.test` but makes use of the **multicore**, **doMC**, and **foreach** packages. Because of this, `maxentMC` only works on POSIX-compliant OS's (essentially anything but Windows).

Examples

```
# example with tussock dataset
traits <- tussock$trait[, c(2:7, 11)] # use only continuous traits
traits <- na.omit(traits) # remove 2 species with NA's
abun <- tussock$abun[, rownames(traits)] # abundance matrix
abun <- t(apply(abun, 1, function(x) x / sum(x) )) # relative abundances
agg <- functcomp(traits, abun) # community-aggregated traits
traits <- t(traits) # transpose matrix

# run maxent on site 1 (first row of abun), all species
pred.abun <- maxent(agg[1,], traits)

## Not run:
# do the constraints give predictive ability?
maxent.test(pred.abun, obs = abun[1,], nperm = 49)

# are height, LDMC, and leaf [N] important constraints?
maxent.test(pred.abun, obs = abun[1,], sub.c = c("height",
"LDMC", "leafN"), nperm = 49)

## End(Not run)
```

`simul.dbFD` *Simulations to Explore Relationships Between Functional Diversity Indices*

Description

`simul.dbFD` generates artificial communities of species with artificial functional traits. Different functional diversity (FD) indices are computed from these communities using `dbFD` to explore their inter-relationships.

Usage

```
simul.dbFD(s = c(5, 10, 15, 20, 25, 30, 35, 40), t = 3,
          r = 10, p = 100, tr.method = c("unif", "norm", "lnorm"),
          abun.method = c("lnorm", "norm", "unif"), w.abun = TRUE)
```

Arguments

`s`	vector listing the different levels of species richness used in the simulations
`t`	number of traits
`r`	number of replicates per species richness level
`p`	number of species in the common species pool
`tr.method`	character string indicating the sampling distribution for the traits. `"unif"` is a uniform distribution, `"norm"` is a normal distribution, and `"lnorm"` is a lognormal distribution.

abun.method	character string indicating the sampling distribution for the species abundances. Same as for tr.method.
w.abun	logical; should FDis, FEve, FDiv, and Rao's quadratic entropy (Q) be weighted by species abundances?

Value

A list contaning the following elements:

results	data frame containing the results of the simulations
traits	matrix containing the traits
abun	matrix containing the abundances
abun.gamma	species abundances from the pooled set of communities
FDis.gamma	FDis of the pooled set of communities
FDis.mean	mean FDis from all communities

FDis.gamma and FDis.mean can be used to explore the set concavity criterion (Ricotta 2005) for FDis.

A graph plotting the results of the simulations is also returned.

Warning

The simulations performed by simul.dbFD can take several hours if length(s) and/or r is large. Run a test with the default parameters first.

Author(s)

Etienne Laliberté <etiennelaliberte@gmail.com> https://www.elaliberte.info/

References

Laliberté, E. and P. Legendre (2010) A distance-based framework for measuring functional diversity from multiple traits. *Ecology* **91**299:305.

Ricotta, C. (2005) A note on functional diversity measures. *Basic and Applied Ecology* **6**:479-486.

See Also

dbFD, the function called in simul.dbFD

Examples

```
# this should take just a few minutes
## Not run:
ex1 <- simul.dbFD(s = c(10, 20, 30, 40, 50), r = 5)
ex1

## End(Not run)
```

tussock *Functional Composition of Short-Tussock Grasslands*

Description

tussock contains data on 16 functional traits measured on 53 vascular plant species from New Zealand short-tussock grasslands. It also contains the relative abundances (percent cover) of these 53 species from 30 8x50-m plots.

Usage

```
tussock
```

Format

tussock is a list of 2 components:

trait data frame of 16 functional traits measured on 53 plant species: growth form (sensu Cornelissen et al. 2003), reproductive plant height (m), leaf dry matter content ($\mathrm{mg\ g^{-1}}$), leaf nitrogen concentration ($\mathrm{mg\ g^{-1}}$), leaf phosphorous concentration ($\mathrm{mg\ g^{-1}}$), leaf sulphur concentration ($\mathrm{mg\ g^{-1}}$), specific leaf area ($\mathrm{m^2\ kg^{-1}}$), nutrient uptake strategy (sensu Cornelissen et al. 2003), Raunkiaer life form, clonality, leaf size ($\mathrm{mm^2}$), primary dispersal mode, seed mass (mg), resprouting capacity, pollination syndrome, and lifespan (an ordinal variable stored as ordered).

abun matrix containing the relative abundances (percent cover) of the 53 species in 30 plots

Details

The functional traits were measured using standardized methodologies (Cornelissen et al. 2003). Each of the 30 experimental plots from which species cover was estimated is 8x50 m. Relative abundances of all vascular plant species were estimated in November 2007. To do so, 20 1x1-m quadrats per plot were randomly positioned along two longitudinal transects and cover of each species was estimated using a modified Braun-Blanquet scale. This data was pooled at the plot scale to yield the percent cover data.

Source

Etienne Laliberté <etiennelaliberte@gmail.com>

https://www.elaliberte.info/

References

Cornelissen, J. H. C., S. Lavorel, E. Garnier, S. Diaz, N. Buchmann, D. E. Gurvich, P. B. Reich, H. ter Steege, H. D. Morgan, M. G. A. van der Heijden, J. G. Pausas and H. Poorter. (2003) A handbook of protocols for standardised and easy measurement of plant functional traits worldwide. *Australian Journal of Botany* **51**:335-380.

Laliberté, E., Norton, D. A. and D. Scott. (2008) Impacts of rangeland development on plant functional diversity, ecosystem processes and services, and resilience. *Global Land Project (GLP) Newsletter* **4**:4-6.

Scott, D. (1999) Sustainability of New Zealand high-country pastures under contrasting development inputs. 1. Site, and shoot nutrients. *New Zealand Journal of Agricultural Research* **42**:365-383.

Index

∗ **CWM**
dbFD, 3
FD-package, 2
functcomp, 12
∗ **FDis**
dbFD, 3
FD-package, 2
∗ **FDiv**
dbFD, 3
FD-package, 2
∗ **FEve**
dbFD, 3
FD-package, 2
∗ **FGR**
dbFD, 3
FD-package, 2
∗ **FRic**
dbFD, 3
FD-package, 2
∗ **New Zealand**
tussock, 26
∗ **community weighted means**
dbFD, 3
FD-package, 2
functcomp, 12
∗ **datagen**
simul.dbFD, 24
∗ **datasets**
dummy, 10
tussock, 26
∗ **distribution**
maxent, 18
maxent.test, 21
∗ **functional composition**
dbFD, 3
FD-package, 2
functcomp, 12
tussock, 26
∗ **functional dispersion**
dbFD, 3
FD-package, 2
fdisp, 10
∗ **functional divergence**
dbFD, 3
FD-package, 2
∗ **functional diversity**
dbFD, 3
FD-package, 2
fdisp, 10
functcomp, 12
simul.dbFD, 24
tussock, 26
∗ **functional evenness**
dbFD, 3
FD-package, 2
∗ **functional group richness**
dbFD, 3
FD-package, 2
∗ **functional identity**
functcomp, 12
∗ **functional richness**
dbFD, 3
FD-package, 2
∗ **functional trait**
dbFD, 3
FD-package, 2
functcomp, 12
tussock, 26
∗ **math**
maxent, 18
maxent.test, 21
∗ **maximum entropy**
maxent, 18
maxent.test, 21
∗ **models**
maxent, 18
maxent.test, 21
∗ **multivariate dispersion**

fdisp, 10
∗ **multivariate**
dbFD, 3
fdisp, 10
functcomp, 12
gowdis, 14
mahaldis, 16
∗ **package**
FD-package, 2
∗ **quadratic entropy**
dbFD, 3
FD-package, 2
∗ **statistical mechanics**
maxent, 18
maxent.test, 21

betadisper, *7*, *9*, *11*, *12*

cailliez, *6*
cascadeKM, *5*, *7*
character, *4*, *13*, *14*

daisy, *16*
dbFD, *2*, 3, *12–14*, *24*, *25*
dist, *4*, *6*, *11*, *15*, *17*
dist.binary, *5*, *6*
divc, *5*, *7*, *9*
dudi.pco, *6*
dummy, 10

factor, *4–6*, *13*, *14*
FD *(*FD-package*)*, 2
FD-package, 2
fdisp, *6*, *9*, 10
functcomp, *5*, *7*, *9*, 12, *21*

gowdis, *2*, *4–6*, *9*, *11*, 14

hclust, *5*

is.euclid, *6*

lingoes, *6*

mahalanobis, *17*
mahaldis, 16
maxent, *2*, 18, *22*, *23*
maxent.test, *20*, *21*, 21

NA, *2*, *4*, *6*, *7*, *11*, *13–15*, *17*
numeric, *4*, *6*, *13*, *14*

ordered, *4*, *6*, *13*, *14*, *26*

quasieuclid, *6*

simul.dbFD, *7*, *9*, *11*, 24

treedive, *9*
tussock, 26