Package 'EcotoneFinder'

July 21, 2025

Type Package

Title Characterising and Locating Ecotones and Communities

Version 0.2.3

Author Antoine Bagnaro

Maintainer Antoine Bagnaro <antoine.bagnaro@wanadoo.fr>

Description Analytical methods to locate and characterise ecotones, ecosystems and environmental patchiness along ecological gradients. Methods are implemented for isolated sampling or for space/time series. It includes Detrended Correspondence Analysis (Hill & Gauch (1980) <doi:10.1007/BF00048870>), fuzzy clustering (De Cáceres et al. (2010) <doi:10.1080/01621459.1963.10500845>), biodiversity indices (Jost (2006) <doi:10.1111/j.2006.0030-1299.14714.x>), and network analyses (Epskamp et al. (2012) <doi:10.18637/jss.v048.i04>) - as well as tools to explore the number of clusters in the data. Functions to produce synthetic ecological datasets are also provided.

Depends R (>= 3.5.0)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports cluster, colorspace, corrplot, e1071, ggplot2, graphics, qgraph, igraph, methods, philentropy, plyr, purrr, reshape, rlang, Rmisc, stats, vegan, vegclust, withr

RoxygenNote 6.1.1

Suggests testthat (>= 2.1.0), knitr, rmarkdown, bookdown

NeedsCompilation no

Repository CRAN

Date/Publication 2021-02-16 23:00:02 UTC

Contents

`arrange.vars` *Re-ordering columns in dataframes:*

Description

Re-ordering columns in dataframes:

Usage

```
arrange.vars(data, vars)
```

Arguments

`data`	dataframe to be ordered
`vars`	nammed vectors of new positions. See details.

Details

This function provides an easy way to re-order the columns of a dataframe. The "vars" parameter must be a nammed numeric vectors with the names corresponding to the targeted columns and the numbers corresponding to their desired new positions.

Value

The dataframe with the desired colum order.

Examples

```
#### Dummy data:
dat <- data.frame("Fac1" = c(rep("A", 6), rep("B",6)),
                  "Var1" = rnorm(12, mean = 20, sd = 1),
                  "Fac2" = rep(c("Low","High","Low","High"),
                               each=3),
                  "Var2" = c(rnorm(3,7), rnorm(3,9),
                             rnorm(3,12), rnorm(3,15)))

# factor columns at the begining.
arrange.vars(dat, vars =c("Fac2" = 2))
# factor columns at the end.
arrange.vars(dat, vars =c("Fac1" = 3, "Fac2" = 4))
```

`cbindna` *qpcR cbind.na method.*

Description

qpcR cbind.na method.

Usage

```
cbindna(..., deparse.level = 1)
```

Arguments

`...`	(generalized) vectors or matrices. See base::cbind
`deparse.level`	integer controlling the construction of labels in the case of non-matrix-like arguments. See base::cbind

Value

a matrix combining the ... arguments column-wise.

Examples

```
### Vectors:
a <- c(rep(1, 5), NA, seq(1:5))
b <- c(rep(1, 4), NA, seq(1:7))

# Complete shorter vector with NAs:
cbindna(a,b)
```

`clustergram` *Clustergram base function*

Description

Clustergram base function

Usage

```
clustergram(Data, k.range = 2:10,
  clustering.function = clustergram.kmeans,
  clustergram.plot = clustergram.plot.matlines, line.width = 0.004,
  add.center.points = TRUE, ...)
```

Arguments

`Data`	Should be a scales matrix. Where each column belongs to a different dimension of the observations
`k.range`	A vector with the number of clusters to plot the clustergram for.
`clustering.function`	Which clustering method to be used. Default is k-means. Can be FCM is set to clustergram.vegclust. See details
`clustergram.plot`	Type of plot for the output. See details.
`line.width`	Graphical parameter. Width of the lines.
`add.center.points`	Logical. Should the cluster means be plotted (as points).
`...`	Additional arguments to be passed to the clustering function.

Details

This is the clustergram function created by Matthias Schonlau. See: Schonlau M. The clustergram: A graph for visualizing hierarchical and nonhierarchical cluster analyses. The Stata Journal. 2002;2:391–402.

It is reproduced in this package for convenience. This package also provide extensions of the clustergram method for fuzzy-c-means clustering and for the evolution of the main fuzzy indices. These

extensions take the form of additional options to be passed in the clustering.function argument and the clustergram.plot argument.

It is also recommended to run the clustergram analysis several times and compare the obtained outputs, as they may vary significantly.

Value

A clustergram plot of the inputed data

Examples

```
####### Example data:
SyntheticTrial <- SyntheticData(SpeciesNum = 100,
                                CommunityNum = 3, SpCo = NULL,
                                Length = 500,
                                Parameters = list(a=c(40, 80, 50),
                                                  b=c(100,250,400),
                                                  c=rep(0.03,3)),
                              dev.c = .015, pal = c("#008585", "#FBF2C4", "#C7522B"))

######## 6 clustergram plots
for (i in 1:6) clustergram(as.matrix(SyntheticTrial[,2:ncol(SyntheticTrial)]),
                           k.range = 2:10, line.width = .2)
```

`clustergram.cmeans` *cmeans function for clustergram*

Description

cmeans function for clustergram

Usage

```
clustergram.cmeans(Data, k, method = "cmeans", ...)
```

Arguments

`Data`	Should be a scales matrix. Where each column belongs to a different dimension of the observations.
`k`	Number of desired groups for the c-means clustering.
`method`	Clustering method for the cmeans function.
`...`	Additional parameters to be passed to the cmeans function.

Details

This is an implementation of Fuzzy c-means clustering (with the cmeans function of the e1071 package) for the clustergram function. The return list is internally used by the clustergram to build the clustergram plot.

Value

A list containing the cluster vector and the centers matrix (see cmeans function).

Examples

```
####### Example data:
   SyntheticTrial <- SyntheticData(SpeciesNum = 100,
                                   CommunityNum = 3, SpCo = NULL,
                                   Length = 500,
                                   Parameters = list(a=c(40, 80, 50),
                                                     b=c(100,250,400),
                                                     c=rep(0.03,3)),
                                 dev.c = .015, pal = c("#008585", "#FBF2C4", "#C7522B"))

   ######## 6 clustergram plots
   for (i in 1:6) clustergram(as.matrix(SyntheticTrial[,2:ncol(SyntheticTrial)]),
                              clustering.function = clustergram.cmeans,
                              k.range = 2:10, line.width = .2)
```

`clustergram.cmeans.Ind` *cmeans clustering with fuzzy indices computation for clustergram*

Description

cmeans clustering with fuzzy indices computation for clustergram

Usage

```
clustergram.cmeans.Ind(Data, k, method = "cmeans", ...)
```

Arguments

`Data`	Should be a scales matrix. Where each column belongs to a different dimension of the observations.
`k`	Number of desired groups for the FCM clustering.
`method`	Clustering method for the cmeans function.
`...`	Additional parameters to be passed to the cmeans function.

Details

Additionally to the FCM clustering with the cmeans function (e1071 package), the function compute the main fuzzy indices to help with the decision on the optimal number of cluster in the data. The indices are computed with the vegclustIndex function of the vegclust package. Maximum values of PCN or minimum values of PEN can be used as criteria to choose the number of clusters.

Value

A list containing the cluster vector, the centers matrix and a vector of four fuzzy indices (partition coefficient (PC), normalized partition coefficient (PCN), partition entropy (PE) and normalized partition entropy (PEN)). See vegclust and veclustIndex functions.

Examples

```
####### Example data:
 SyntheticTrial <- SyntheticData(SpeciesNum = 100,
                                 CommunityNum = 3, SpCo = NULL,
                                 Length = 500,
                                 Parameters = list(a=c(40, 80, 50),
                                                   b=c(100,250,400),
                                                   c=rep(0.03,3)),
                                dev.c = .015, pal = c("#008585", "#FBF2C4", "#C7522B"))

 ######## clustergram plots with fuzzy indices plots:
 clustergramInd(as.matrix(SyntheticTrial[,2:ncol(SyntheticTrial)]),
                               clustering.function = clustergram.cmeans.Ind,
                               clustergram.plot = clustergram.plot.matlines,
                               FuzzyIndice.plot = FuzzyIndice.plot.matlines,
                               k.range = 2:10, line.width = .2)
```

`clustergram.kmeans` *Type function that clustergram takes for clustering.*

Description

Type function that clustergram takes for clustering.

Usage

```
clustergram.kmeans(Data, k, ...)
```

Arguments

`Data`	Should be a scales matrix. Where each column belongs to a different dimension of the observations
`k`	Number of desired groups for the k-means clustering.
`...`	Additional parameters to be passed in the kmeans function (from the stats package).

Details

This is the type of function that the clustergram function uses for clustering. The return list is internally used by the clustergram to build the clustergram plot.

Value

A list containing the cluster vector and the centers matrix (see kmeans function).

Examples

```
####### Example data:
 SyntheticTrial <- SyntheticData(SpeciesNum = 100,
                                 CommunityNum = 3, SpCo = NULL,
                                 Length = 500,
                                 Parameters = list(a=c(40, 80, 50),
                                                   b=c(100,250,400),
                                                   c=rep(0.03,3)),
                               dev.c = .015, pal = c("#008585", "#FBF2C4", "#C7522B"))

 ######## 6 clustergram plots
 for (i in 1:6) clustergram(as.matrix(SyntheticTrial[,2:ncol(SyntheticTrial)]),
                            clustering.function = clustergram.kmeans,
                            k.range = 2:10, line.width = .2)
```

`clustergram.plot.matlines` *Plot function for clustergram*

Description

Plot function for clustergram

Usage

```
clustergram.plot.matlines(X, Y, k.range, x.range, y.range, COL,
  add.center.points, centers.points)
```

Arguments

`X`	vector of the k number of cluster for the x-axis
`Y`	corrdinates of the cluster centers on the y-axis
`k.range`	x axis breaks.
`x.range`	x axis range.
`y.range`	y axis range (PCA scores).
`COL`	colour palette.
`add.center.points`	Should the centers be plotted as points. (see clustergram)
`centers.points`	matrix of centers position.

Details

Internal clustergram plot function. The input arguments are computed by the clustergram function directly.

Value

A clustergram plot.

Examples

```
####### Example data:
   SyntheticTrial <- SyntheticData(SpeciesNum = 100,
                                   CommunityNum = 3, SpCo = NULL,
                                   Length = 500,
                                   Parameters = list(a=c(40, 80, 50),
                                                     b=c(100,250,400),
                                                     c=rep(0.03,3)),
                                 dev.c = .015, pal = c("#008585", "#FBF2C4", "#C7522B"))

   ######## 6 clustergram plots
   for (i in 1:6) clustergram(as.matrix(SyntheticTrial[,2:ncol(SyntheticTrial)]),
                              clustering.function = clustergram.kmeans,
                              clustergram.plot = clustergram.plot.matlines,
                              k.range = 2:10, line.width = .2)
```

---

clustergram.vegclust *Vegclust function for clustergram*

Description

Vegclust function for clustergram

Usage

```
clustergram.vegclust(Data, k, method = method, ...)
```

Arguments

Data	Should be a scales matrix. Where each column belongs to a different dimension of the observations.
k	Number of desired groups for the FCM clustering.
method	Clustering method for the vegclust function.
...	Additional parameters to be passed to the vegclust function.

Details

This is an implementation of Fuzzy c-means clustering (with the vegclust function of the vegclust package) for the clustergram function. The return list is internally used by the clustergram to build the clustergram plot.

Value

A list containing the cluster vector and the centers matrix (see vegclust function).

Examples

```
####### Example data:
 SyntheticTrial <- SyntheticData(SpeciesNum = 100,
                                 CommunityNum = 3, SpCo = NULL,
                                 Length = 500,
                                 Parameters = list(a=c(40, 80, 50),
                                                   b=c(100,250,400),
                                                   c=rep(0.03,3)),
                                dev.c = .015, pal = c("#008585", "#FBF2C4", "#C7522B"))

 ######## clustergram plot
 clustergram(as.matrix(SyntheticTrial[,2:ncol(SyntheticTrial)]),
                            clustering.function = clustergram.vegclust,
                            k.range = 2:10, line.width = .2)
```

`clustergram.vegclust.Ind`

Vegclust clustering with fuzzy indices computation for clustergram

---

Description

Vegclust clustering with fuzzy indices computation for clustergram

Usage

```
clustergram.vegclust.Ind(Data, k, method = "FCM", ...)
```

Arguments

`Data`	Should be a scales matrix. Where each column belongs to a different dimension of the observations.
`k`	Number of desired groups for the FCM clustering.
`method`	Clustering method for the vegclust function.
`...`	Additional parameters to be passed to the vegclust function.

Details

Additionally to the FCM clustering, the function compute the main fuzzy indices to help with the decision on the optimal number of cluster in the data. Maximum values of PCN or minimum values of PEN can be used as criteria to choose the number of clusters.

Value

A list containing the cluster vector, the centers matrix and a vector of four fuzzy indices (partition coefficient (PC), normalized partition coefficient (PCN), partition entropy (PE) and normalized partition entropy (PEN)). See vegclust and veclustIndex functions.

Examples

```
####### Example data:
 SyntheticTrial <- SyntheticData(SpeciesNum = 100,
                                 CommunityNum = 3, SpCo = NULL,
                                 Length = 500,
                                 Parameters = list(a=c(40, 80, 50),
                                                   b=c(100,250,400),
                                                   c=rep(0.03,3)),
                                dev.c = .015, pal = c("#008585", "#FBF2C4", "#C7522B"))

 ######## clustergram plots with fuzzy indices plots:
 clustergramInd(as.matrix(SyntheticTrial[,2:ncol(SyntheticTrial)]),
                                clustering.function = clustergram.vegclust.Ind,
                                clustergram.plot = clustergram.plot.matlines,
                                FuzzyIndice.plot = FuzzyIndice.plot.matlines,
                                k.range = 2:10, line.width = .2)
```

`clustergramInd` *Clustergram with fuzzy indices plot*

Description

Clustergram with fuzzy indices plot

Usage

```
clustergramInd(Data, k.range = 2:10,
  clustering.function = clustergram.kmeans,
  clustergram.plot = clustergram.plot.matlines,
  FuzzyIndice.plot = FuzzyIndice.plot.matlines, line.width = 0.004,
  add.center.points = TRUE, ...)
```

Arguments

`Data`	Should be a scales matrix. Where each column belongs to a different dimension of the observations.
`k.range`	A vector with the number of clusters to plot the clustergram for.
`clustering.function`	Which clustering method to be used. Default is k-means. Can be FCM is set to clustergram.vegclust. See details
`clustergram.plot`	Type of plot for the clustergram output. See details.
`FuzzyIndice.plot`	Type of plot for the fuzzy indices output. See details.
`line.width`	Graphical parameter. Width of the lines.
`add.center.points`	Logical. Should the cluster means be plotted (as points).
`...`	Additional arguments to be passed to the clustering function.

Details

This clustergram fuction produces an additional plot with the evolution of the main fuzzy indices (normalized partition coefficient (PCN) and normalized partition entropy (PEN)). Maximum values of PCN or minimum values of PEN can be used as criteria to choose the number of clusters.

Value

A clustergram plot and a fuzzy indices evolution plot of the inputed data

Examples

```
####### Example data:
   SyntheticTrial <- SyntheticData(SpeciesNum = 100,
                                   CommunityNum = 3, SpCo = NULL,
                                   Length = 500,
                                   Parameters = list(a=c(40, 80, 50),
                                                     b=c(100,250,400),
                                                     c=rep(0.03,3)),
                                  dev.c = .015, pal = c("#008585", "#FBF2C4", "#C7522B"))

   ######## clustergram plots with fuzzy indices plots:
   clustergramInd(as.matrix(SyntheticTrial[,2:ncol(SyntheticTrial)]),
                                  clustering.function = clustergram.vegclust.Ind,
                                  clustergram.plot = clustergram.plot.matlines,
                                  FuzzyIndice.plot = FuzzyIndice.plot.matlines,
                                  k.range = 2:10, line.width = .2)
```

`CommunityColor` *Tool to assign color to species distribution plots given fuzzy clustering results.*

Description

Tool to assign color to species distribution plots given fuzzy clustering results.

Usage

```
CommunityColor(ecotonefinder, method = c("vegclust", "cmeans"),
  pal = c("diverge_hcl", "terrain_hcl", "sequential_hcl", "rainbow_hcl"))
```

Arguments

`ecotonefinder`	A list containing elements named in the same way than EcotoneFinder function outcomes. Must contain at least one of vegclust or cmeans results.
`method`	Which fuzzy clustering to use. Either vegclust or cmeans.
`pal`	Which palette to use for color picking. Chosen from the colorspace package.

Details

CommunityColor creates a color vector that can be used by plotting functions. It assigns colors to species of a community matrix given the results of vegclust or cmeans analyses (using cmeans$centers or vegclust$mobileCenters). Species are assigned to a color according to the cluster centroid for which they have their highest membership value (see Bandelj et al., 2012).

The palette must be one of the colorspace package.

Value

A vector of color names from the palette in the pal argument, of the same length and in the same order than the species columns of the provided data.

Examples

```
######## Artificial dataset & analysis:
SyntheticTrial <- SyntheticData(SpeciesNum = 27, CommunityNum = 3,
                                SpCo = NULL, Length = 500,
                                Parameters = list(a=rep(60, 3),
                                                  b=c(0,250,500),
                                                  c=rep(0.03,3)),
                                dev.c = .015,
                                pal = c("#008585", "#FBF2C4", "#C7522B"))

SyntheticEcoFinder <- EcotoneFinder(SyntheticTrial[,-1],
                                    dist = SyntheticTrial$Distance,
                                    method = "all", groups = 3,
                                    standardize = "hellinger",
```

```
                                    diversity = "all")

##### Assigning colors to communities:
SyntheticColor <- CommunityColor(SyntheticEcoFinder, pal = "diverge_hcl",
                  method = "cmeans")

#### Plotting:
plotEcotone(data = SyntheticEcoFinder, plot.data = TRUE, plot.method = "none",
            col.data = SyntheticColor)
```

---

curveNoPlot *Adaptation of the curve function (without plot).*

Description

Adaptation of the curve function (without plot).

Usage

```
curveNoPlot(expr, from = NULL, to = NULL, n = 101, add = FALSE,
  type = "l", xname = "x", xlab = xname, ylab = NULL, log = NULL,
  xlim = NULL, ...)
```

Arguments

expr	The name of a function, or a call or an expression written as a function of x which will evaluate to an object of the same length as x.
from	the range over which the function will be plotted (start).
to	the range over which the function will be plotted (end).
n	integer; the number of x values at which to evaluate.
add	logical; if TRUE add to an already existing plot; if NA start a new plot taking the defaults for the limits and log-scaling of the x-axis from the previous plot. Taken as FALSE (with a warning if a different value is supplied) if no graphics device is open.
type	plot type: see plot.default.
xname	character string giving the name to be used for the x axis.
xlab	labels and graphical parameters.
ylab	labels and graphical parameters.
log	labels and graphical parameters. See 'Details' for the interpretation of the default for log.
xlim	NULL or a numeric vector of length 2; if non-NULL it provides the defaults for c(from, to) and, unless add = TRUE, selects the x-limits of the plot – see plot.window.
...	Additional graphical arguments.

Details

Silently used by SyntheticData and SyntheticDataSeries. Equivalent to the curve function of the graphics package. See the details of the curve function in graphics package for more details.

Value

A vector containing the y values of the gaussian along the gradient.

Examples

```
gaussian <- function(x) a*exp(-(((x-b)^2)/2*(c^2)))
a <- 60
b <- 250
c <- 0.4
Curve=curveNoPlot(gaussian, from = 1, to = 500, n = 500)
Curve$y
```

DistEco *Tools for internal data structure exploration*

Description

Tools for internal data structure exploration

Usage

```
DistEco(data, distance.method = "inner_product", transpose = TRUE,
  symm = FALSE, plot = c("heatmap", "network"),
  palette = "colorblind", spinglass = TRUE, run = 10,
  spinglass.groups = c("rounded", "raw"), manual.groups = NULL,
  return.network = TRUE, ...)
```

Arguments

`data`	A community or environmental matrix, containing species or variables as columns and sites as rows.
`distance.method`	The distance method to be used for the calculation of the distance matrix. Must be one of philentropy::distance
`transpose`	Logical. If TRUE, the distance matrix is calculated between species. If FALSE, it is calculated between sites.
`symm`	Logical indicating if x should be treated symmetrically; can only be true when x is a square matrix. See stats::heatmap.
`plot`	The kind of plot produced by the function. Can be “heatmap” or “network”.

palette	The colour palette for the network, if spinglass = TRUE. Must be one of the palettes supported by qgraph. Default to "colorblind".
spinglass	Logical. Whether or not to run a spinglass algorithm to produce statistical groups for the network. The spinglass algorithm is performed with the CommunityNetwork function.
run	Number of runs for the spinglass algorithm. Higher numbers produce more trustable results but rapidly increase computation time. Default to 10.
spinglass.groups	If spinglass = TRUE, the type of grouping to use from the results of the spinglass algorithm. See Details.
manual.groups	If spinglass = FALSE, an object that indicates which nodes belong together. Can be a list in which each element is a vector of integers identifying the numbers of the nodes that belong together, or a factor.
return.network	Logical. If TRUE, the qgraph object is returned as output of the function.
...	Additionnal parameters for heatmap or qgraph.

Value

A plot corresponding to the plot argument.

Examples

```
### Artificial data:
SyntheticTrial <- SyntheticData(SpeciesNum = 21, CommunityNum = 3,
                                SpCo = NULL, Length = 500,
                                Parameters = list(a=rep(60, 3),
                                                  b=c(0,250,500),
                                                  c=rep(0.01,3)),
                                pal = c("#008585", "#FBF2C4", "#C7522B"))

## Network of species, with raw spinglass groups:
DistEco(SyntheticTrial[,2:ncol(SyntheticTrial)], transpose = TRUE,
        plot = c("network"), spinglass = TRUE, run = 10,
        spinglass.groups = c("raw"))

## Heatmap of species:
DistEco(SyntheticTrial[,2:ncol(SyntheticTrial)], transpose = TRUE,
        symm = FALSE, plot = c("heatmap"))
```

EcotoneFinder *Wraper function to perform ecological gradient analysis*

Description

Wraper function to perform ecological gradient analysis

Usage

```
EcotoneFinder(data, dist, method = c("dca", "fanny", "vegclust",
  "diversity", "cmeans", "all"), groups = NULL, m.exp = 2,
  standardize = NULL, seed = 1, diversity = c("shannon", "richness",
  "expShannon", "pielou", "all"), na.rm = FALSE)
```

Arguments

data	A dataframe containing species as columns and sites as rows. May also contain environmental parameters, as long as the parameters are as columns and the site as rows.
dist	A vector or column containing the gradient along which the analysis will be done. Must be of the same lenght as data.
method	One of c("dca", "fanny", "vegclust", "diversity", "cmeans", "all"). Tell the function which analysis to perform. See details.
groups	Interger. The desired number of clusters if any of the clustering method is selected.
m.exp	Integer. The membership exponent for any of the clustering method.
standardize	Standardize method to apply to the data before further analysis (for fanny and cmeans). Must be one of decostand methods (see decostand).
seed	Integer or NULL. Set a seed for initial membership matrix for cmeans and vegclust algorithms. Recomended for time series, to keep a more consistent labelling of fuzzy clusters along the gradient. See Details.
diversity	diversity indice to be calculated. See details.
na.rm	Logical. Should NAs be removed.

Details

EcotoneFinder is a wraper function to perform multiple ecological gradient analysis at once. The implemented methods are Detrended Correspondance Analysis (DCA) - see Brownstein et al. 2013 - Fuzzy C-Means (FCM) - see DeCaceres et al., 2010 - and the calculation of diversity indices - see Jost, 2007. The DCA is intenally performed by the decorana function of the vegan package. The FCM analyses can be performed by the fanny function (cluster package), the vegclust function (vegclust package) and the cmeans function (e1071 package) - for comparison purposes as the outcome of the analyses might differ. The vegclust and cmeans algorithms use random number generators to create a matrix of initial centers. Setting a seed with the *seed* argument guaranty the reproducibility of the outputs. This argument can also be set to NULL, to preserve the randomness of initial centres.

It is recommended to stadardize data before applying fanny or cmeans analysis. See decostand documentation (vegan package) for information on standardisation methods. Must be one of: "total","max","freq","normalize","range","pa","chi.square","hellinger","log". If "log" is chosen, the user will be asked to provide the base to be used upon launching the function.

Several diversity incides have been implemented, as they are supposed to react to ecological gradients. It includes the Shannon index, the Pielou eveness and species richness - computed with the diversity function of the vegan package. expShannon corresponds to the shannon index in terms of

effective number of species (see Jost, 2007). If "all" is selected, all the implemented indices will be calculated.

If "all" is selected in the method argument, all the implemented methods will be applied.

Value

Ecofinder returns a list containing the original data, the value of the main arguments used in the function and the outcome of the selected analyses.

Examples

```
#### Artificial dataset:
SyntheticTrial <- SyntheticData(SpeciesNum = 21, CommunityNum = 3,
                                SpCo = NULL, Length = 500,
                                Parameters = list(a=rep(60, 3),
                                                  b=c(0,250,500),
                                                  c=rep(0.015,3)),
                                pal = c("#008585", "#FBF2C4", "#C7522B"))

## Analyses:
SyntheticEcoFinder <- EcotoneFinder(data = SyntheticTrial[,-1],
                                    dist = SyntheticTrial$Distance,
                                    method = "all",
                                    groups = 3, standardize = "hellinger",
                                    diversity = "all")
```

EcotoneFinderSeries *Extension of EcotoneFinder for space/time series*

Description

Extension of EcotoneFinder for space/time series

Usage

```
EcotoneFinderSeries(data, dist, series = NULL, method = c("dca",
  "fanny", "vegclust", "diversity", "cmeans", "all"), groups = NULL,
  m.exp = 2, standardize = NULL, diversity = c("shannon", "richness",
  "expShannon", "pielou", "all"), na.rm = FALSE)
```

Arguments

data	A list of dataframes corresponding to the series or a single dataframe with a series factor. Species must appear as columns.
dist	A vector or column containing the gradient along which the analysis will be done. Must be of the same lenght as data.
series	If data is a single dataframe, must be the name or the number of the factor column identifying the series.
method	One of c("dca", "fanny", "vegclust", "diversity", "cmeans", "all"). Tell the function which analysis to perform. See details.
groups	Interger. The desired number of clusters if any of the clustering method is selected.
m.exp	Integer. The membership exponent for any of the clustering method.
standardize	Standardize method to apply to the data before further analysis (for fanny and cmeans). Must be one of decostand methods (see decostand).
diversity	diversity indice to be calculated. See details.
na.rm	Logical. Should NAs be removed.

Details

EcotoneFinderSeries is a generalisation of the EcotoneFinder function to handle space/time series of data. If a dataframe is provided, it will convert it internally to a named list according to the factor provided by the series argument. The methods of analysis and standardizations, as well as the diversity indices are the same as those of the EcotoneFinder function.

Value

A list of lists containing the outcomes of the EcotoneFinder function for each series.

Examples

```
############# Synthetic time series data:
SyntheticTrialSeries <- SyntheticDataSeries(CommunityPool = 40,
                                            CommunityNum = 4, SpCo = NULL,
                                            Length = 500, SeriesNum = 5,
                                            Parameters = list(a=rep(60, 4),
                                                              b=c(0,200,350,500),
                                                              c=rep(0.03,4)),
                                            dev.c = .015,
                                       pal = c("#008585", "#B8CDAE", "#E6C186", "#C7522B"),
                                            replacement = FALSE,
                                            Parameters.repl = TRUE)

EcoTimeSeriesTrial <- EcotoneFinderSeries(data = SyntheticTrialSeries,
                                          dist = "Distance", method = "cmeans",
                                          series = "Time", groups = 4,
                                          standardize = "hellinger", na.rm = TRUE)
```

`ExtractCentroid` *Visualisation of fuzzy centroids:*

Description

Visualisation of fuzzy centroids:

Usage

```
ExtractCentroid(ecotonefinder, method = c("fanny", "cmeans", "vegclust"),
  normalized = c("species", "cluster", "none"), position = "dodge",
  threshold = 0, plot = TRUE, col = NULL, return.plot = TRUE,
  labels = ggplot2::waiver(), main = "Community composition",
  xlab = "species", ylab = "Centroid contribution", cex.x = 12)
```

Arguments

`ecotonefinder`	A list containing elements named in the same way than EcotoneFinder function outcomes. Must contain "cmeans", "fanny" or "vegclust" results.
`method`	The fuzzy clustering results from which the centroids will be extracted.
`normalized`	Method to normalise the centroid values, either by "species" or "cluster". If "none", the centroids are plotted without transformation. See details.
`position`	Set the positions of the bars for the barchart. This is passed down to the geom_bar function of ggplot. Default is set to "dodge".
`threshold`	Threshold for centroid contribution value under which the species will not be plotted. Can be used to simplify plots containing many species. See Details.
`plot`	Logical. Should the plot be displayed. If FALSE, the centroids matrix is returned without plotting.
`col`	Colour vector for the plot. Should be of the same length that the number of fuzzy clusters.
`return.plot`	Logical. Should the GGplot object be stored internally (e.g. for multi-ploting). Default is TRUE.
`labels`	Character vectors of labels for the legend. Must be of the same length that the number of fuzzy clusters.
`main`	Main title for the plot. See plot.
`xlab`	A title for the x-axis. See plot.
`ylab`	A title for the y-axis. See plot.
`cex.x`	cex for the x-axis labels.

Details

This function extracts and plots the fuzzy centroids species contributions, according to user-defined normalisation steps and threshold value. The contributions of the different species in the fuzzy centroids may be used as a proxy for community compositions. The cmeans function (cmeans package) and vegclust function (vegclust package) internally compute the centroid compositions and their outputs are directly used by the ExtractCentroid function. The fanny function (cluster package), however, does not provide internal centroids calculation. They are computed here as:

$$Centroid[clusterj] = \sum[ij](Membership[ij]xObservation[ij])/\sum[j]Membership[j]$$

Where the centroid of a cluster is the mean of all observations, weighted by their degree of belonging to the cluster. The obtained species contributions to the centroids of the fuzzy clusters can then be plotted as they are, if normalised = "none". To obtain more intuitive units for the interpretation of the species contributions, two normalisation methods are proposed. If normalised = "cluster", the species contributions are given in percent per clusters (i.e. the sum of all species contributions in each cluster centroid equals 100). If normalised = "species", each species has its contributions summed to 100 (i.e. each species is in percent per cluster). For normalised = "none" and normalised = "cluster", a threshold value can be specified. Species that do not score above this threshold will not be displayed on the resulting plot. This can be used to simplify the outputs, for dataset containing large number of species.

Value

A matrix containing the cluster centroids.

Examples

```
##### Artificial dataset & analyses:
 SyntheticTrial <- SyntheticData(SpeciesNum = 20, CommunityNum = 4,
                                 SpCo = NULL ,Length = 500,
                                 Parameters=list(a = rep(60, 4),
                                                 b = c(0,150,350,500),
                                                 c = rep(0.015,4)),
                                 dev.c = 0.007,
                                 pal = c("#008585", "#B8CDAE", "#E6C186", "#C7522B"))

 EcoFinder <- EcotoneFinder(SyntheticTrial[,-1],
                            dist = SyntheticTrial$Distance,
                            method = "all", groups=4,
                            standardize = "hellinger",
                            diversity="all")

 ##### Centroid plot without normalisation:
 Centroid <- ExtractCentroid(EcoFinder, method = "fanny",
                             normalized = "none",  threshold = 0,
                             plot = TRUE, position = "dodge",
                             col = colorspace::heat_hcl(4))

 ##### Centroid plot normalised by clusters:
 Centroid <- ExtractCentroid(EcoFinder, method = "fanny",
                             normalized = "cluster",  threshold = 0,
```

```
                                plot = TRUE, position = "dodge",
                                col = colorspace::heat_hcl(4))
```

`FuzzyIndice.plot.matlines` *Plot function for fuzzy indices with clustergram.*

Description

Plot function for fuzzy indices with clustergram.

Usage

```
FuzzyIndice.plot.matlines(Z, k.range, x.range, z.range)
```

Arguments

`Z`	Fuzzy indices matrix.
`k.range`	x axis breaks.
`x.range`	x axis range.
`z.range`	y axis range for the fuzzy indices.

Details

This function provide the tools to add a fuzzy indices evolution plot together with the normal clustegram plot with the evolution of the relative positions of the cluster centers.

Value

A plot with the evolution of the fuzzy indices given the number of fuzzy clusters that were applied to the data.

Examples

```
####### Example data:
   SyntheticTrial <- SyntheticData(SpeciesNum = 100,
                                   CommunityNum = 3, SpCo = NULL,
                                   Length = 500,
                                   Parameters = list(a=c(40, 80, 50),
                                                     b=c(100,250,400),
                                                     c=rep(0.03,3)),
                                  dev.c = .015, pal = c("#008585", "#FBF2C4", "#C7522B"))

   ######## clustergram plots with fuzzy indices plots:
   clustergramInd(as.matrix(SyntheticTrial[,2:ncol(SyntheticTrial)]),
                                  clustering.function = clustergram.vegclust.Ind,
                                  clustergram.plot = clustergram.plot.matlines,
```

```
                                FuzzyIndice.plot = FuzzyIndice.plot.matlines,
                                k.range = 2:10, line.width = .2)
```

---

ggEcotone *GGplot method for EcotoneFinder*

Description

GGplot method for EcotoneFinder

Usage

```
ggEcotone(ecotonefinder, slope = NULL, plot.data = FALSE,
  method = c("none", "dca", "fanny", "vegclust", "cmeans", "diversity",
  "dca_slope", "fanny_slope", "vegclust_slope", "cmeans_slope",
  "diversity_slope"), axis.number = 1, diversity = c("Shannon",
  "SpeciesRichness", "ExpShannon", "Pielou", "SpeciesRichness_slope",
  "Shannon_slope", "ExpShannon_slope", "Pielou_slope"), facet = NULL,
  col = "black", title = NULL, xlab = NULL, ylab = NULL,
  return.plot = TRUE)
```

Arguments

ecotonefinder	list containing elements named in the same way than the EcotoneFinder function outcomes.
slope	list containing elements named in the same way than the Slope function outcomes.
plot.data	Logical. Should the data be plotted? Default to FALSE.
method	Analysis method to be plotted from the EcotoneFinder results or the Slope results. Must be one or several of "none","dca","fanny","vegclust", "cmeans","diversity", "dca_slope","fanny_slope","vegclust_slope", "cmeans_slope" or "diversity_slope".
axis.number	Number of DCA axis to be plotted. Must be between 1 and 4. Default to 1.
diversity	diversity indice to be plotted, if the method argument contains "diversity" or "diversity_slope". Must be one or several of "Shannon", "SpeciesRichness", "ExpShannon", "Pielou", "all"
facet	Character vector of method names indicating how the plot should be facetted. Can be provided as a list if several methods are to be plotted on the same facets. Can contain "data" if plot.data = T. If NULL, no facets are returned. See details.
col	Color palette to be used for plotting. Must be either of length 1, of the same lenght than the number of facets (when provided), or of the same length than the number of species (if plot.data = TRUE), or than the number of groups or axis plotted with the method argument. See details.
title	Main title for the plot

xlab	A title for the x-axis. See plot.
ylab	A title for the y-axis. See plot.
return.plot	Logical. If TRUE, the plot is directly plotted. If FALSE, the plot is stored as a ggplot object. Default to FALSE. See details.

Details

The ggEcotone function is intended to facilitate the plotting of EcotoneFinder lists with the use of the ggplot2 grammar. It either directly print its outputs (if plot = TRUE), or returns a ggplot object that can be further modified (if plot = FALSE). The latter allows for the addition of other ggplot2 layers to personalise graphical outputs (see examples).

Facetting options are implemented to allow for the separation of the different method outputs and facilitate comparisons. The facet parameter accepts lists, with each element of the list corresponding to a facet and consisting of the names of the methods to be plotted on that facet.

The col parameter allows for basic control over the colors of the lines. ggplot internally recycles colour vectors for each new facets, making it difficult to precisely control colours in facetted plots. Plotting the outputs on several graphs and arranging them on a grid is the best way to produce "facetted" plots with different coulour schemes. See examples.

Value

A ggplot object.

Examples

```
#### Artificial dataset:
 SyntheticTrial <- SyntheticData(SpeciesNum = 21, CommunityNum = 3,
                                 SpCo = NULL ,Length = 500,
                                 Parameters = list(a=rep(60, 3),
                                                   b=c(0,250,500),
                                                   c=rep(0.015,3)),
                                 pal = c("#008585", "#FBF2C4", "#C7522B"))

 ## Analyses:
 EcoFinder <- EcotoneFinder(data = SyntheticTrial[,-1],
                            dist = SyntheticTrial$Distance,
                            method = "all", groups = 3,
                            standardize = "hellinger", diversity = "all")

 ## Slope calculation:
 EcoSlope <- Slope(EcoFinder, method = "all", axis.number = 2,
                   diversity = "all")

 ## Plots:

 require(ggplot2)
 require(colorspace)
 # Species Distributions and Fuzzy clusters:
 Plot <- ggEcotone(EcoFinder, slope = EcoSlope, plot.data = TRUE,
                   method = c("cmeans", "fanny"),
```

```
                  col = c("#D33F6A", "#E99A2C", "#E2E6BD"),
                  facet = list(c("data"), c("cmeans", "fanny")),
                  title = "Species distribution and fuzzy clusters",
                  xlab = "Gradient", ylab = "Membership grades") +
  theme(plot.title = element_text(hjust = 0.5, face="bold")) +
  theme_bw()
Plot

# Fuzzy clusters & derivatives:
Plot <- ggEcotone(EcoFinder, slope = EcoSlope, plot.data = FALSE,
                  method = c("cmeans", "cmeans_slope"),
                  col = c("#D33F6A", "#E99A2C", "#E2E6BD"),
                  facet = c("cmeans", "cmeans_slope"),
                  title = "fuzzy clusters and derivatives",
                  xlab = "Gradient", ylab = "Membership grades") +
  theme(plot.title = element_text(hjust = 0.5, face="bold")) +
  theme_bw()
Plot

# Multiplot layout:
GG1 <- ggEcotone(EcoFinder, slope = EcoSlope, plot.data = TRUE,
                 method = c("none"), col = heat_hcl(21), facet = NULL,
                 title = "Species distributions", xlab = NULL,
                 ylab = "Abundances") +
  theme(plot.title = element_text(hjust = 0.5, face="bold")) +
  theme_bw()

GG2 <- ggEcotone(EcoFinder, slope = EcoSlope, plot.data = FALSE,
                 method = c("cmeans"), col = c("#023FA5", "#BEC1D4", "#D6BCC0"),
                 facet = NULL, title = "Fuzzy clusters", xlab = NULL,
                 ylab = "Membership grades") +
  theme(plot.title = element_text(hjust = 0.5, face="bold")) +
  theme_bw()

GG3 <- ggEcotone(EcoFinder, slope = EcoSlope, plot.data = FALSE,
                 method = c("diversity"),
                 col = c("#26A63A", "#B4B61A"), facet = NULL,
                 diversity=c("SpeciesRichness", "ExpShannon"),
                 title = "diversity indices", xlab = "Gradient",
                 ylab = "Index scores") +
  theme(plot.title = element_text(hjust = 0.5, face="bold")) +
  theme_bw()

require(Rmisc)
Rmisc::multiplot(GG1,GG2,GG3)
```

NetworkCommunity *Perform Spinglass algorythm and find networks communities*

Description

Perform Spinglass algorythm and find networks communities

Usage

```
NetworkCommunity(networkeco, run = 100)
```

Arguments

networkeco	A network object (either qgraph or igraph) or a list created by the NetworkEco or NetworkEcoSeries functions.
run	Number of runs for the spinglass algorithm. Computation may be heavy for high numbers.

Details

The function perform spinglass algorithm on the provided network. (see spinglass.community() function of the igraph package for more details) The provided graph is internally transformed into a igraph object if needed. The function returns a number of summary statistics from the n runs of the spinglass algorithm. Each run of the spinglass algorithm is done with a different seed, to ensure different outputs. The seeds are internally recycled by the *with_seed* fuction of the *withr* package, so that the global environment is not modified. The frequencies at which a number of communities are recognised in the network and the average assignements (rounded or not) of the nodes into these communities are returned by the function. The latter can help to statistically define groups for network graphical representations.

Value

A list containing the number of runs, the number of possible communities defined by the spinglass algorithm (with frequencies) and the mean and rounded mean of the assignement of the nodes of the network to these communities.

Examples

```
#### Artificial data:
SyntheticTrial <- SyntheticData(SpeciesNum = 21, CommunityNum = 3,
                                SpCo = NULL, Length = 500,
                                Parameters = list(a=rep(60, 3),
                                                  b=c(0,250,500),
                                                  c=rep(0.01,3)),
                                pal = c("#008585", "#FBF2C4", "#C7522B"))

# Building first network:
Network <- DistEco(SyntheticTrial[,2:ncol(SyntheticTrial)],
                   transpose = TRUE, plot = c("network"), spinglass = FALSE,
                   return.network = TRUE)

### Spinglass algorithm (increase number of run for better accuracy):
SpinglassTrial <- NetworkCommunity(Network, run = 5)
```

```
### Network with spinglass groups:
DistEco(SyntheticTrial[,2:ncol(SyntheticTrial)], transpose = TRUE,
        plot = c("network"), spinglass = FALSE, return.network = FALSE,
        manual.groups = as.factor(SpinglassTrial$Memberships$RoundedMean))
```

NetworkEco *Networks for ecotones and communities*

Description

Networks for ecotones and communities

Usage

```
NetworkEco(ecotonefinder, threshold = 0.8, plot.type = c("percentage",
  "corrplot", "heatmap", "network"), method = c("cmeans", "vegclust"),
  dist.method = "inner_product", plot = c("species", "community"),
  order.sp = NULL, dist = c("count", "relative", "raw"),
  no.plot = FALSE, network.group = NULL, ...)
```

Arguments

ecotonefinder	A list containing elements named in the same way than EcotoneFinder function outcomes. Must contain cmeans results or vegclust results.
threshold	If count = T, the membership grade threshold used to sort the species in the different clusters.
plot.type	Which graphical representation to be plotted. Among "percentage", "corrplot", "heatmap","network".
method	The membership computation method to be used. One of "cmeans" or "vegclust". Must be present in the ecotonefinder list.
dist.method	Distance method for the computation of a distance matrix, when dist = "raw" and dist = "relative".
plot	If plot = "species", the distances are computed between the species in the data. If plot = "community", the distances are computed between the cluster centroids.
order.sp	Vector providing the order in which to arrange the species. If NULL, the column order will be kept.
dist	The type of data on which distance calculations are made from. If dist = "raw", the distance matrix is computed from the membership matrix directly. if dist = "relative", the distance matrix is computed from the relative memberships grades of each species in the clusters (between 0 and 1). If dist = "count", the species are assigned to clusters according to the threshold and the distance matrix is computed from the number of common species between the different clusters. See details.

no.plot	Logical. Should the plot be displayed?. Set to TRUE to gain computation time with large community matrix.
network.group	Grouping parameter for the network. Can be user defined (see qgraph documentation for details) but must be a factor of the same lenght as the nodes of the graph.
...	Additional arguments to be passed to the plotting functions, see details.

Details

The NetworkEco function provides wqys to explore the relations between fuzzy clusters. Several options are implemented. If dist = "raw", it computes a distance matrix from the membership grade matrix directly. If dist = "relative", the membership grades are standardized so that the sum of the membership grades of a given species equals to 1 for every points along the gradient (which corresponds to a percentage ot membership in each cluster). If dist = "count", the standardized membership grades of the are used to assign species in the community to a unique cluster and the number of common species between pairs of clusters is counted. The assignement of species to clusters is done by listing all the species that score a membership grade higher than the specified threshold in a cluster. The resulting list of species are then compared to one another.

The function also allows the computation of distances between species rather than between clusters, when plot = "species". This can only be done from the memberships grades (raw or relative) and this argument will be disregarded if dist = "count".

Several methods of visualisation are implemented: "percentage", "corrplot", "heatmap" and "network". If "percentage", a barplot (using ggplot2) of the standardized memberships grades per fuzzy cluster is plotted. It always plot the standardized membership grades regardless of the chosen dist option, but if dist = "count" or dist = "raw" are chosen, the function still compute the corresponding distance matrices and return them ti the output list. For time efficiency, it is not recommended to plot it when the number of species in the community is large (>100). "corrplot" and "heatmap" produce correlation matrix and heat map. The "network" is based on the qgraph function of the qgraph package. The ... argument may be used to pass additional arguments to the plotting functions (for graphical purposes).

Value

A list containing the percentage matrix, the distance matrix and the network object (depending of the arguments passed to the function)

Examples

```
#### Artificial dataset:
SyntheticTrial <- SyntheticData(SpeciesNum = 21, CommunityNum = 3,
                                SpCo = NULL, Length = 500,
                                Parameters = list(a=rep(60, 3),
                                                  b=c(0,250,500),
                                                  c=rep(0.015,3)),
                                pal = c("#008585", "#FBF2C4", "#C7522B"))

## Analyses:
SyntheticEcoFinder <- EcotoneFinder(data = SyntheticTrial[,-1],
                                    dist = SyntheticTrial$Distance,
```

```
                                    method = "all",
                                    groups = 3, standardize = "hellinger",
                                    diversity = "all")

## Percentage plot:
SyntheticNetwork <- NetworkEco(SyntheticEcoFinder, threshold = .3, method = "cmeans",
                               plot.type = "percentage", dist = "count")

## Heatmap plot:
SyntheticNetwork <- NetworkEco(SyntheticEcoFinder, plot.type = "heatmap",
                               method = "cmeans", dist = "raw", plot = "species")

## Network:
# From raw membership grades:
SyntheticNetwork <- NetworkEco(SyntheticEcoFinder, plot.type = "network",
                               method = "cmeans", dist = "raw", plot = "species")

# From number of species per clusters:
SyntheticNetwork <- NetworkEco(SyntheticEcoFinder, plot.type = "network", threshold = .3,
                               method = "cmeans", dist = "count", plot = "community",
                               layout = "spring")
```

NetworkEcoSeries *Networkeco for data series*

Description

Networkeco for data series

Usage

```
NetworkEcoSeries(ecotonefinder, threshold = 0.8, method = c("cmeans",
  "vegclust"), plot.type = c("percentage", "heatmap", "corrplot",
  "network"), plot = c("species", "community"), no.plot = FALSE,
  order.sp = NULL, dist.method = "inner_product", dist = c("count",
  "relative", "raw"), network.group = c("site", "cluster"),
  method.corr = "number", ...)
```

Arguments

ecotonefinder	A list containing elements named in the same way than EcotoneFinderSeries function outcomes. Must contain cmeans results or vegclust results.
threshold	If dist = "count", the membership grade threshold used to sort the species in the different clusters.
method	The membership computation method to be used. Must be present in the ecotonefinder list.

plot.type	Which graphical representation to be plotted. Among "percentage", "corrplot", "heatmap","network"
plot	If plot = "species", the distances are computed between the species in the data. If plot = "community", the distances are computed between the cluster centroids.
no.plot	Logical. Should the plot be displayed?. Set to TRUE to gain computation time with large community matrix.
order.sp	Vector providing the order in which to arrange the species. If NULL, the column order will be kept.
dist.method	Distance method for the computation of a distance matrix, when dist = "raw" or dist = "percent".
dist	The type of data on which distance calculations are made from. If dist = "raw", the distance matrix is computed from the membership matrix directly. if dist = "relative", the distance matrix is computed from the relative memberships grades of each species in the clusters (between 0 and 1). If dist = "count", the species are assigned to clusters according to the threshold and the distance matrix is computed from the number of common species between the different clusters. See details.
network.group	If network.group = "site" the nodes of the networks will be colored according to the different times or sites of the series. If network.group = "cluster" the nodes of the network will be colored according to the different fuzzy clusters. Can be user defined (see qgraph documentation for details) but must be a factor of the same lenght as the nodes of the graph.
method.corr	If plot.type = "corrplot", the method to be used for the corrplot. Must be one of "circle", "square", "ellipse", "number", "shade", "color", "pie". Default to "number".
...	Additional arguments to be passed to the plotting functions, see details.

Details

NetworkEcoSeries is a generalisation of the NetworkEco function to analyses space/time series. The ... argument may be used to pass additional arguments to the plotting functions (for graphical purposes).

Value

A list containing the percentage matrix, the distance matrix and the network object (depending of the arguments passed to the function)

Examples

```
SyntheticTrialSeries <- SyntheticDataSeries(CommunityPool = 40,
                                            CommunityNum = 4, SpCo = NULL,
                                            Length = 500, SeriesNum = 5,
                                            Parameters = list(a=rep(60, 4),
                                                              b=c(0,200,350,500),
                                                              c=rep(0.03,4)),
                                    pal = c("#008585", "#B8CDAE", "#E6C186", "#C7522B"),
```

```
                                          replacement = TRUE,
                                          Parameters.repl = TRUE)

  EcoTimeSeriesTrial <- EcotoneFinderSeries(data = SyntheticTrialSeries,
                                        dist = "Distance",
                                        method = c("cmeans","vegclust"),
                                        series = "Time", groups = 4,
                                        standardize = "hellinger", na.rm=TRUE)

#### Network from the common number of species above membership threshold between clusters:
  SyntheticNetworkSeries <- NetworkEcoSeries(EcoTimeSeriesTrial, threshold = .2,
                                         method = "cmeans", plot.type = "network",
                                         plot = "community", dist = "count",
                                         network.group = "cluster",
                                         dist.method = "inner_product",
                                         no.plot = FALSE, layout = "spring",
                                         shape = "ellipse",
                                         palette = "colorblind")

#### Network of relations between species from their raw membership values in each cluster:
  SyntheticNetworkSeries <- NetworkEcoSeries(EcoTimeSeriesTrial, threshold = .2,
                                         method = "cmeans", plot.type = "network",
                                         plot = "species", dist = "raw",
                                         dist.method = "inner_product",
                                         no.plot = FALSE, layout = "spring",
                                         shape = "ellipse",
                                         palette = "colorblind")
```

---

plotEco *Plotting component for EcotoneFinder*

Description

Plotting component for EcotoneFinder

Usage

```
plotEco(ecotonefinder, plot.data = FALSE, plot.method = c("none",
  "dca", "fanny", "vegclust", "cmeans", "diversity"), axis.number = 1,
  magnification = 20, magnification.diversity = 5,
  col.data = "black", col.method = c("red", "blue"), title = NULL,
  ylab = "Species", xlab = "Gradient", na.rm = FALSE, alone = TRUE,
  ...)
```

Arguments

ecotonefinder	A list containing elements named in the same way than EcotoneFinder function outcomes

plot.data	Logical. Should the data be plotted.
plot.method	Analysis method to be plotted from the EcotoneFinder analyses. Must be one or several of "none","dca","fanny","vegclust", "cmeans" or"diversity".
axis.number	Number of axis to plot from the DCA.
magnification	Magnification coefficient for the method. Usefull if the data are being plotted.
magnification.diversity	Particular magnification for the diversity indices.
col.data	Colors to be used for the data. See CommunityColor function.
col.method	Colors to be used for the methods.
title	An overall title for the plot. See plot.
ylab	A title for the y-axis. See plot.
xlab	A title for the x-axis. See plot.
na.rm	Logical. Should NAs be removed.
alone	Logical. If FALSE, lines are added to an existing plot.
...	Additional argument to be passed to the plot function.

Details

Internal component of the PlotEcotone function for the plotting of the EcotoneFinder analyses. Use PlotEcotone directly for more options.

Value

A plot with the EcotoneFinder results along the gradient, and optionally, the data.

Examples

```
######## Artificial dataset & analysis:
SyntheticTrial <- SyntheticData(SpeciesNum = 20, CommunityNum = 3,
                                SpCo = NULL, Length = 500,
                                Parameters = list(a=rep(60, 3),
                                                  b=c(0,250,500),
                                                  c=rep(0.03,3)),
                                dev.c = .015, pal = c("#008585", "#FBF2C4", "#C7522B"))

SyntheticEcoFinder <- EcotoneFinder(SyntheticTrial[,-1],
                                    dist = SyntheticTrial$Distance,
                                    method = "all", groups = 3,
                                    standardize = "hellinger",
                                    diversity = "all")

### Plot:
require(colorspace)
plotEco(SyntheticEcoFinder, plot.data = FALSE,
        plot.method = c("cmeans", "dca"),
        axis.number = 2, col.method = terrain_hcl(3))
```

plotEcotone *Plot method for EcotoneFinder*

Description

Plot method for EcotoneFinder

Usage

```
plotEcotone(data = NULL, slope = NULL, env = NULL,
  plot.data = FALSE, plot.method = c("none", "dca", "fanny",
  "vegclust", "cmeans", "diversity", "dca_slope", "fanny_slope",
  "vegclust_slope", "cmeans_slope", "diversity_slope"), axis.number = 1,
  magnification = 20, magnification.diversity = 5,
  magnification.slope = 500, col.data = "black",
  col.method = c("red", "blue"), col.slope = c("darkgreen", "green"),
  col.env = c("orange", "gold"), title = NULL, ylab = "Species",
  xlab = "Gradient", na.rm = FALSE, ...)
```

Arguments

data	A list containing elements named in the same way than EcotoneFinder function outcomes
slope	A list containing elements named in the same way than Slope function outcomes
env	A list containing elements named in the same way than EcotoneFinder function outcomes. Usefull if EcotoneFinder has been run on environmental data and the outcomes are to be compared with the outcomes from the community matrix. Can also be used to compare results from two different community matrices, if the x-axis are similar.
plot.data	Logical. Should the data be plotted.
plot.method	Analysis method to be plotted from the EcotoneFinder results or the Slope results. Must be one or several of "none","dca","fanny","vegclust", "cmeans","diversity", "dca_slope","fanny_slope","vegclust_slope", "cmeans_slope" or "diversity_slope".
axis.number	Number of axis to plot from the DCA.
magnification	Magnification coefficient for the method. Usefull if the data are being plotted.
magnification.diversity	Particular magnification for the diversity indices.
magnification.slope	Magnification coefficient for the Slope.
col.data	Colors to be used for the data. See CommunityColor function.
col.method	Colors to be used for the methods (for data).
col.slope	Colors to be used for the methods (for slope).
col.env	Colors to be used for the methods (for env).

title	An overall title for the plot. See plot.
ylab	A title for the y-axis. See plot.
xlab	A title for the x-axis. See plot.
na.rm	Logical. Should NAs be removed.
...	Additional argument to be passed to the plot function.

Details

The plotEcotone function is intended for easy visualisation of the results of the EcotoneFinder function and the Slope function along the sampling gradient. It also provide a way to plot the original species data - for comparison - along with magnification coefficients for the different method or slopes in order to facilitate visualization. Please note that large sets of species may be confusing to plot. The CommunityColor function is also provided to help with data coloring and easy visualisation.

Value

A plot corresponding to the plotEcotone arguments.

Examples

```
######## Artificial dataset & analysis:
SyntheticTrial <- SyntheticData(SpeciesNum = 20, CommunityNum = 3,
                                SpCo = NULL, Length = 500,
                                Parameters = list(a=rep(60, 3),
                                                  b=c(0,250,500),
                                                  c=rep(0.03,3)),
                                dev.c = .015, pal = c("#008585", "#FBF2C4", "#C7522B"))

SyntheticEcoFinder <- EcotoneFinder(SyntheticTrial[,-1],
                                    dist = SyntheticTrial$Distance,
                                    method = "all", groups = 3,
                                    standardize = "hellinger",
                                    diversity = "all")

##### Assigning colors to communities:
SyntheticColor <- CommunityColor(SyntheticEcoFinder, pal = "diverge_hcl",
                                 method = "cmeans")

###### Computing the derivatives:
SyntheticSlope <- Slope(SyntheticEcoFinder, method = "all",
                        axis.number = 2, diversity = "all")

####### Plot the derivative of the FCM with the synthetic species data:
require(colorspace)
plotEcotone(slope = SyntheticSlope, plot.data = TRUE,
            plot.method = c("cmeans_slope"), axis.number = 2,
            col.method = terrain_hcl(3), col.data = SyntheticColor)

####### Plot the derivative and the FCM:
```

```
require(colorspace)
plotEcotone(data = SyntheticEcoFinder, slope = SyntheticSlope,
            plot.data = TRUE,
            plot.method = c("cmeans", "cmeans_slope"),
            axis.number = 2, col.method = terrain_hcl(3),
            col.data = SyntheticColor)
```

---

plotEnv *Plotting component for EcotoneFinder when run on environmental data*

Description

Plotting component for EcotoneFinder when run on environmental data

Usage

```
plotEnv(env, plot.data = FALSE, plot.method = c("none", "dca", "fanny",
  "vegclust", "cmeans", "diversity"), axis.number = 1,
  magnification = 20, magnification.diversity = 5,
  col.data = "black", col.method = c("red", "blue"), title = NULL,
  ylab = "Species", xlab = "Gradient", na.rm = FALSE, alone = TRUE,
  ...)
```

Arguments

env	A list containing elements named in the same way than EcotoneFinder function outcomes.
plot.data	Logical. Should the data be plotted.
plot.method	Analysis method to be plotted from the EcotoneFinder analyses. Must be one or several of "none","dca","fanny","vegclust", "cmeans" or"diversity".
axis.number	Number of axis to plot from the DCA.
magnification	Magnification coefficient for the method. Usefull if the data are being plotted.
magnification.diversity	Particular magnification for the diversity indices.
col.data	Colors to be used for the data. See CommunityColor function.
col.method	Colors to be used for the methods.
title	An overall title for the plot. See plot.
ylab	A title for the y-axis. See plot.
xlab	A title for the x-axis. See plot.
na.rm	Logical. Should NAs be removed.
alone	Logical. If FALSE, lines are added to an existing plot.
...	Additional argument to be passed to the plot function.

Details

Internal component of the PlotEcotone function for the plotting of the EcotoneFinder analyses. Use PlotEcotone directly for more options. The "diversity" method is still implemented, but will send a warning as it may not be relevant for environmental data.

Value

A plot with the EcotoneFinder results along the gradient, and optionally, the data.

Examples

```
######## Artificial dataset & analysis:
SyntheticTrial <- SyntheticData(SpeciesNum = 20, CommunityNum = 3,
                                SpCo = NULL, Length = 500,
                                Parameters = list(a=rep(60, 3),
                                                  b=c(0,250,500),
                                                  c=rep(0.03,3)),
                                dev.c = .015, pal = c("#008585", "#FBF2C4", "#C7522B"))

SyntheticEcoFinder <- EcotoneFinder(SyntheticTrial[,-1],
                                    dist = SyntheticTrial$Distance,
                                    method = "all", groups = 3,
                                    standardize = "hellinger",
                                    diversity = "all")

### Plot:
require(colorspace)
plotEnv(SyntheticEcoFinder, plot.data = FALSE,
        plot.method = c("cmeans", "dca"),
        axis.number = 2, col.method = terrain_hcl(3))
```

`plotSlope` *Plotting component for Slope*

Description

Plotting component for Slope

Usage

```
plotSlope(ecotoneslope, plot.data = FALSE, plot.method = c("none",
  "dca_slope", "fanny_slope", "vegclust_slope", "cmeans_slope",
  "diversity_slope"), axis.number = 1, magnification = 500,
  col.data = "black", col.method = c("red", "blue"), title = NULL,
  ylab = "Species", xlab = "Gradient", na.rm = FALSE, alone = TRUE,
  ...)
```

Arguments

ecotoneslope	A list containing elements named in the same way than Slope function outcomes
plot.data	Logical. Should the data be plotted.
plot.method	Analysis method to be plotted from the Slope results. Must be one or several of "none", "dca_slope","fanny_slope","vegclust_slope", "cmeans_slope" or "diversity_slope".
axis.number	Number of axis to plot from the DCA.
magnification	Magnification coefficient for the method. Usefull if the data are being plotted.
col.data	Colors to be used for the data. See CommunityColor function.
col.method	Colors to be used for the methods.
title	An overall title for the plot. See plot.
ylab	A title for the y-axis. See plot.
xlab	A title for the x-axis. See plot.
na.rm	Logical. Should NAs be removed.
alone	Logical. If FALSE, lines are added to an existing plot.
...	Additional argument to be passed to the plot function.

Details

Internal component of the PlotEcotone function for the plotting of the Slope analyses. Use PlotEcotone directly for more options.

Value

A plot with the Slope results along the gradient, and optionally, the data.

Examples

```
######## Artificial dataset & analysis:
SyntheticTrial <- SyntheticData(SpeciesNum = 20, CommunityNum = 3,
                                SpCo = NULL, Length = 500,
                                Parameters = list(a=rep(60, 3),
                                                  b=c(0,250,500),
                                                  c=rep(0.03,3)),
                                dev.c = .015, pal = c("#008585", "#FBF2C4", "#C7522B"))

SyntheticEcoFinder <- EcotoneFinder(SyntheticTrial[,-1],
                                    dist = SyntheticTrial$Distance,
                                    method = "all", groups = 3,
                                    standardize = "hellinger",
                                    diversity = "all")

### Derivatives:
SyntheticSlope <- Slope(SyntheticEcoFinder, method = "all",
                        axis.number = 2, diversity = "all")
```

```
### Plot:
require(colorspace)
plotSlope(SyntheticSlope, plot.data = FALSE,
          plot.method = c("cmeans_slope", "vegclust_slope"),
          col.method = terrain_hcl(3))
```

`rbindna` *qpcR rbind.na method.*

Description

qpcR rbind.na method.

Usage

```
rbindna(..., deparse.level = 1)
```

Arguments

`...`	(generalized) vectors or matrices. See base::rbind
`deparse.level`	integer controlling the construction of labels in the case of non-matrix-like arguments. See base::rbind

Value

a matrix combining the ... arguments row-wise.

Examples

```
### Vectors:
a <- c(rep(1, 5), NA, seq(1:5))
b <- c(rep(1, 4), NA, seq(1:7))

# Complete shorter vector with NAs:
rbindna(a,b)
```

`Slope` *Method to calculate the derivative of irregular functions:*

Description

Method to calculate the derivative of irregular functions:

Usage

```
Slope(ecotonefinder, method = c("dca", "fanny", "vegclust", "cmeans",
  "diversity", "all"), window = 3, axis.number = 1,
  groups = ecotonefinder$groups, diversity = c("shannon", "richness",
  "expShannon", "pielou", "all"))
```

Arguments

`ecotonefinder`	A list containing elements named in the same way than EcotoneFinder function outcomes
`method`	The name of the method for which the slopes should be calculated. Correspond to the names of the list.
`window`	Must be an odd number. The interval to be used for slope calculation. The bigger the window, the more averaged the slope will be.
`axis.number`	If "dca" is chosen, indicate the number of axis over which to calculate the slope (first axis, first and second axis,...)
`groups`	If any clustering method is chosen, corresponds to the index of the cluster for which the slope shold be calculated. If "all", the slope will be calculated for all the clusters.
`diversity`	If "diversity" is chosen in the method argument, define the diversity index for which to calculate the slope. "all" can be chosen.

Details

Slope calculations are done by moving window analysis. The width of the windows is defined by the window argument. For each window, the result of slope coefficient of a linear model (lm function of the stat package) is stored and used to draw the general slope along the gradient. The bigger the window, the more points will be used to compute the linear models, meaning the obtained slopes will be smoother. This also results in the addition of NAs at the ends of the gradient.

The first axis of DCA has been used as a beta-diversity index, and its derivative as a method to locate ecotones (see Brownstein et al., 2013). The Slope function provide the possibility of computing the slope of the other axis, to avoid the loss of information induced by the reduction of the dimentionality of the original data. Similarly, the slopes of the fuzzy clusters can be used to pinpoint the transitions between them. The value of the slopes can be an indicator of the relative sharpness of the transion area. Particularly, as the memberships of the fuzzy clusters range betwwen 0 and 1, these values can readilly be compared between studies and datasets. These values vary depending on the window width and can be very sensible to noise in the original data. A reliable method to mathematically identify breaks is still needed and careful interpretation by the user is still required.

Value

A list of dataframes containing the slope values for the specified methods and the original data.

Examples

```
#### Artificial dataset:
SyntheticTrial <- SyntheticData(SpeciesNum = 21, CommunityNum = 3,
                                SpCo = NULL ,Length = 500,
                                Parameters = list(a=rep(60, 3),
                                                  b=c(0,250,500),
                                                  c=rep(0.015,3)),
                                pal = c("#008585", "#FBF2C4", "#C7522B"))

## Analyses:
SyntheticEcoFinder <- EcotoneFinder(data = SyntheticTrial[,-1],
                                    dist = SyntheticTrial$Distance,
                                    method = "all", groups = 3,
                                    standardize = "hellinger", diversity = "all")

## Slope calculation:
SyntheticSlope <- Slope(SyntheticEcoFinder, method = "all", axis.number = 2,
                        diversity = "all")
```

SyntheticData *Create synthetic gaussian-shaped species abundance data*

Description

Create synthetic gaussian-shaped species abundance data

Usage

```
SyntheticData(SpeciesNum, CommunityNum, Length = 100, SpCo = NULL,
  Parameters = list(a = NULL, b = NULL, c = NULL), dev.a = 10,
  dev.b = 10, dev.c = 0, down.limit = 1, pal = NULL,
  xlab = "Gradient", ylab = "Synthetic species",
  title = "Synthetic data")
```

Arguments

SpeciesNum	An integer giving the total number of species in the synthetic data.
CommunityNum	An integer giving the number of communities to be synthetised.
Length	The lenght of the gradient. Corresponds to the x-axis in a plot.

SpCo	The ratio of species per communities. If NULL, species will be spread evenly between communities with additional species in the last community if the quotient is not an integer. When specified, SpCo must be a vector of lenght equal to CommunityNum and whose sum is equal to SpeciesNum
Parameters	A list containing the parameters (a, b and c) for the gaussians. Each parameter must be specified for each community. See Details.
dev.a	The deviation around parameter a for the gaussian in a community. If 0 all species curve in the comunity will have the same a parameter.
dev.b	The deviation around parameter b for the gaussian in a community. If 0 all species curve in the comunity will have the same b parameter.
dev.c	The deviation around parameter a for the gaussian in c community. If 0 all species curve in the comunity will have the same c parameter.
down.limit	The limit under which the gaussian curve will be rounded down to 0. The default is 1.
pal	The color palette to be used. Species curves are colored according to communities. Either a colorspace palette or a vector of the same lenght as the number of species.
xlab	A title for the x-axis. See plot.
ylab	A title for the y-axis. See plot.
title	An overall title for the plot. See plot.

Details

The SyntheticData function is intended for the creation of articficial dataset to test ecological patterns along gradients. The gaussian curves that it computes are of the form: $a*exp(-(((x-b)^2)/2*(c^2)))$ The parameters can be interpreted as follow: a is the maximum height of the gaussian on the y-axis, b is the center of the gaussian on the x-axis and c is the steepness of the slopes on each side of the maximum. The gaussians create a set of continuous data that are akin to abundances. As gaussians of this type cannot reach 0, any value that is below the down.limit (default is 1) is rounded down to 0.

Value

SyntheticData returns a dataset with numbered species (sp.1, sp.2, ...) as columns. It also plot the obtained data.

Examples

```
### 3 distinct communities comprising a total of 21 species
SyntheticTrial <- SyntheticData(SpeciesNum = 21, CommunityNum = 3,
                                SpCo = NULL, Length = 500,
                                Parameters = list(a=rep(60, 3),
                                                  b=c(0,250,500),
                                                  c=rep(0.03,3)),
                                pal = c("#008585", "#FBF2C4", "#C7522B"))

### 3 distinct communities with uneven species numbers
```

```
SyntheticTrial <- SyntheticData(SpeciesNum = 21, CommunityNum = 3,
                                SpCo = c(5, 10, 6), Length = 500,
                                Parameters = list(a=rep(60, 3),
                                                  b=c(0,250,500),
                                                  c=rep(0.03,3)),
                                pal = c("#008585", "#FBF2C4", "#C7522B"))
```

SyntheticDataSeries *Synthetic data for Space/Time series*

Description

Synthetic data for Space/Time series

Usage

```
SyntheticDataSeries(CommunityPool, CommunityNum, Length = 100,
  SpCo = NULL, SeriesNum, replacement = TRUE,
  range.repl = as.integer(CommunityPool/5), Parameters = list(a = NULL,
  b = NULL, c = NULL), Parameters.repl = TRUE, dev.a = 10,
  dev.b = 10, dev.c = 0, Parameters.range = 10,
  displacement = NULL, pal = NULL, xlab = "Gradient",
  ylab = "Synthetic species", title = "Synthetic data")
```

Arguments

CommunityPool	Total number of species per series. must either be of length 1 (if the pool of species is of the same length for all communities) or of length equal to CommunityNum (specifying the species pool for each community)
CommunityNum	An integer giving the number of communities.
Length	The lenght of the gradient. Corresponds to the x-axis in a plot.
SpCo	The ratio of species per communities. When replacement = T, SpCo is computed from CommunityPool and range.repl.
SeriesNum	The number of series to be synthetised.
replacement	Logical. Should the species of a community in a given series be a sample of the possible number of species in this community. See Details.
range.repl	An integer. Gives the possible range of species turnover in a community.
Parameters	A list containing the parameters (a, b and c) for the gaussians. Each parameter must be specified for each community. See Details.
Parameters.repl	Logical. Sould the parameters vary between series.

`dev.a` The deviation around parameter a for the gaussian in a community. If 0 all species curve in the comunity will have the same a parameter.

`dev.b` The deviation around parameter b for the gaussian in a community. If 0 all species curve in the comunity will have the same b parameter.

`dev.c` The deviation around parameter a for the gaussian in c community. If 0 all species curve in the comunity will have the same c parameter.

`Parameters.range` An integer. Gives the possible range of the parameter variations between two series. The range of variation is the parameter diveded by the parameters range. Default is 10.

`displacement` Numeric matrix to control the direction of the changes in the mean values of the parameters over the series. See examples.

`pal` The color palette to be used. Species curves are colored according to communities.

`xlab` A title for the x-axis. See plot.

`ylab` A title for the y-axis. See plot.

`title` An overall title for the plot. See plot.

Details

SyntheticDataSeries is a extention of the SyntheticData function and intended to produce easy and consistent space/time series of artificial ecological community datasets. The series of dataframes are stored in an object of class list. The replacement and Parameters.repl arguments allow the user to choose wether or not the number of species and their distribution curves should vary among the different series. range.repl is an integer that define the boundaries of the interval in which the number of species can vary - such as the number of species per community is : *sample([[CommunityPool/CommunityNum - range.repl ; CommunityPool/CommunityNum + range.repl]])* The number it takes is then used to define SpCo. Given the difference of scales between the 3 parameters a, b and c, the Parameters.range argument controls the variations of the parameters by divison so that they correspond to : *sample([[Parameters - Parameters/Parameters.range ; Parameters + Parameters/Parameters.range]])* The obtained parameters are then used by the internal SyntheticData function as base parameters for a given series, on which dev.a, dev.b and dev.c will apply. The other arguments are equivalent to those of SyntheticData.

Value

SyntheticDataSeries returns a list of datasets with numbered species (sp.1, sp.2, ...) as columns. The list has length = SeriesNum. It also plot the obtained data.

Examples

```
##### 5 datasets of 40 species spread on 4 communities without turnover
##### on the number of species nor variations in their distribution:

SyntheticTrialSeries <- SyntheticDataSeries(CommunityPool = 40,
                                            CommunityNum = 4, SpCo = NULL,
                                            Length = 500, SeriesNum = 5,
```

```
                                        Parameters = list(a=rep(60, 4),
                                                          b=c(0,200,350,500),
                                                          c=rep(0.03,4)),
                                    pal = c("#008585", "#B8CDAE", "#E6C186", "#C7522B"),
                                        replacement = FALSE,
                                        Parameters.repl = FALSE)

##### 5 datasets of 40 species spread on 4 communities with species turnover
##### and variations in their distributions along the gradient:

SyntheticTrialSeries <- SyntheticDataSeries(CommunityPool = 40,
                                        CommunityNum = 4, SpCo = NULL,
                                        Length = 500, SeriesNum = 5,
                                        Parameters = list(a=rep(60, 4),
                                                          b=c(0,200,350,500),
                                                          c=rep(0.03,4)),
                                    pal = c("#008585", "#B8CDAE", "#E6C186", "#C7522B"),
                                        replacement = TRUE,
                                        Parameters.repl = TRUE)

##### With a displacement matrix to control the direction of the changes
##### between series:

# Dispalcement matrix (Parameters x Communities):
disp <- matrix(data=c(0,0,0,
                      0,35,-0.0007,
                      0,10,0), nrow = 3, ncol = 3)

Series <- SyntheticDataSeries(CommunityPool = 60, CommunityNum = 3, Length = 500,
                              SeriesNum = 5, replacement = FALSE, SpCo = c(15,15,30),
                              Parameters = list(a = c(60,60,60),
                                                b = c(-50,-50,400),
                                                c = c(0.01, 0.01, 0.01)),
                              dev.a=30, dev.b=40, dev.c=0,
                              displacement = disp,
                              pal = c(rep("#008585",15), rep("#FBF2C4",15),
                                      rep("#C7522B",30)))
```

Index