

Package ‘BiodiversityR’

November 23, 2014

Type Package

Title GUI for biodiversity, suitability and community ecology analysis

Version 2.5-1

Date 2014-11-20

Author Roeland Kindt

Maintainer Roeland Kindt <R.KINDT@CGIAR.ORG>

Description This package provides a GUI (Graphical User Interface, via the R-Commander) and some utility functions (often based on the vegan package) for statistical analysis of biodiversity and ecological communities, including species accumulation curves, diversity indices, Renyi profiles, GLMs for analysis of species abundance and presence-absence, distance matrices, Mantel tests, and cluster, constrained and unconstrained ordination analysis. A book on biodiversity and community ecology analysis is available for free download from the website. In 2012, methods for (ensemble) suitability modelling and mapping were expanded in the package.

License GPL-2

URL <http://www.r-project.org>,
<http://www.worldagroforestry.org/resources/databases/tree-diversity-analysis>

Depends R (>= 3.0.0), tcltk, vegan (>= 2.2-0)

Imports Rcmdr (>= 2.1-4)

Suggests permute, lattice, MASS, mgcv, cluster, car, RODBC, rpart, effects, multcomp, ellipse, maptree, sp, splanets, spatial, akima, nnet, dismo, raster (>= 2.0-31), rgdal, gbm, randomForest, gam, earth, mda, kernlab, e1071, tools, methods, bootstrap

NeedsCompilation no

Repository CRAN

Date/Publication 2014-11-23 08:33:47

R topics documented:

BiodiversityR-package	3
accumresult	4
add.spec.scores	6
balanced.specaccum	7
BCI.env	9
BiodiversityRGUI	10
CAPdiscrim	33
caprescale	35
crosstabanalysis	37
deviancepercentage	38
dist.eval	39
dist.zeroes	40
distdisplayed	41
disttransform	43
diversityresult	44
ensemble.batch	46
ensemble.dummy.variables	55
ensemble.raster	58
ensemble.test	64
evaluation.strip.data	78
famea	81
ifri	82
importancevalue	83
loaded.citations	85
makecommunitydataset	86
multiconstrained	87
nested.anova.dbrda	89
NMSrandom	90
nnetrandom	92
ordicoeno	93
ordisymbol	94
PCAsignificance	96
radfitresult	98
rankabundance	99
removeNAcomm	101
renyiresult	103
residualssurface	105
spatialsample	107
transfgradient	108
transfspecies	109
warcom	110
warenv	116

Description

This package provides a GUI (Graphical User Interface, via the R-Commander; [BiodiversityRGUI](#)) and some utility functions (often based on the `vegan` package) for statistical analysis of biodiversity and ecological communities, including species accumulation curves, diversity indices, Renyi profiles, GLMs for analysis of species abundance and presence-absence, distance matrices, Mantel tests, and cluster, constrained and unconstrained ordination analysis. A book on biodiversity and community ecology analysis is available for free download from the website.

Details

We warmly thank all that provided inputs that lead to improvement of the Tree Diversity Analysis manual that describes common methods for biodiversity and community ecology analysis and its accompanying software. We especially appreciate the comments received during training sessions with draft versions of this manual and the accompanying software in Kenya, Uganda and Mali. We are equally grateful to the thoughtful reviews by Dr Simoneta Negrete-Yankelevich (Instituto de Ecologia, Mexico) and Dr Robert Burn (Reading University, UK) of the draft version of this manual, and to Hillary Kipruto for help in editing of this manual. We also want to specifically thank Mikkel Grum, Jane Poole and Paulo van Breugel for helping in testing the packaged version of the software. We also want to give special thanks for all the support that was given by Jan Beniest, Tony Simons and Kris Vanhoutte in realizing the book and software.

We highly appreciate the support of the Programme for Cooperation with International Institutes (SII), Education and Development Division of the Netherlands Ministry of Foreign Affairs, and VVOB (The Flemish Association for Development Cooperation and Technical Assistance, Flanders, Belgium) for funding the development for this manual. We also thank VVOB for seconding Roeland Kindt to the World Agroforestry Centre (ICRAF). The tree diversity analysis manual was inspired by research, development and extension activities that were initiated by ICRAF on tree and landscape diversification. We want to acknowledge the various donor agencies that have funded these activities, especially VVOB, DFID, USAID and EU.

We are grateful for the developers of the R Software for providing a free and powerful statistical package that allowed development of BiodiversityR. We also want to give special thanks to Jari Oksanen for developing the `vegan` package and John Fox for developing the `Rcmdr` package, which are key packages that are used by BiodiversityR.

Author(s)

Maintainer: Roeland Kindt (World Agroforestry Centre)

References

Kindt, R. & Coe, R. (2005) Tree diversity analysis: A manual and software for common statistical methods for ecological and biodiversity studies.

<http://www.worldagroforestry.org/resources/databases/tree-diversity-analysis>

We suggest to use this citation for this software as well (together with citations of all other packages that were used)

accumresult

Alternative Species Accumulation Curve Results

Description

Provides alternative methods of obtaining species accumulation results than provided by functions `specaccum` and `plot.specaccum` (**vegan**).

Usage

```
accumresult(x,y="", factor="", level, scale="", method="exact", permutations=100,
  conditioned=T, gamma="boot", ...)
accumplot(xr, addit=F, labels="", col=1, ci=2, pch=1, type="p", cex=1, xlim=c(1, xmax),
  ylim=c(1, rich), xlab="sites", ylab="species richness", ...)
accumcomp(x,y="", factor, scale="", method="exact", permutations=100,
  conditioned=T, gamma="boot", plotit=T, labelit=T, legend=T, rainbow=T,
  xlim=c(1, max), ylim=c(0, rich), type="p", xlab="sites",
  ylab="species richness", ...)
```

Arguments

x	Community data frame with sites as rows, species as columns and species abundance as cell values.
y	Environmental data frame.
factor	Variable of the environmental data frame that defines subsets to calculate species accumulation curves for.
level	Level of the variable to create the subset to calculate species accumulation curves.
scale	Continuous variable of the environmental data frame that defines the variable that scales the horizontal axis of the species accumulation curves.
method	Method of calculating the species accumulation curve (as in function <code>specaccum</code>). Method "collector" adds sites in the order they happen to be in the data, "random" adds sites in random order, "exact" finds the expected (mean) species richness, "coleman" finds the expected richness following Coleman et al. 1982, and "rarefaction" finds the mean when accumulating individuals instead of sites.
permutations	Number of permutations to calculate the species accumulation curve (as in function <code>specaccum</code>).
conditioned	Estimation of standard deviation is conditional on the empirical dataset for the exact SAC (as in function <code>specaccum</code>).
gamma	Method for estimating the total extrapolated number of species in the survey area (as in <code>specaccum</code>).

<code>addit</code>	Add species accumulation curve to an existing graph.
<code>xr</code>	Result from <code>specaccum</code> or <code>accumresult</code> .
<code>col</code>	Colour for drawing lines of the species accumulation curve (as in function <code>plot.specaccum</code>).
<code>labels</code>	Labels to plot at left and right of the species accumulation curves.
<code>ci</code>	Multiplier used to get confidence intervals from standard deviation (as in function <code>plot.specaccum</code>).
<code>pch</code>	Symbol used for drawing the species accumulation curve (as in function <code>points</code>).
<code>type</code>	Type of plot (as in function <code>plot</code>).
<code>cex</code>	Character expansion factor (as in function <code>plot</code>).
<code>xlim</code>	Limits for the horizontal axis.
<code>ylim</code>	Limits for the vertical axis.
<code>xlab</code>	Label for the horizontal axis.
<code>ylab</code>	Label for the vertical axis.
<code>plotit</code>	Plot the results.
<code>labelit</code>	Label the species accumulation curves with the levels of the categorical variable.
<code>legend</code>	Add the legend (you need to click in the graph where the legend needs to be plotted).
<code>rainbow</code>	Use rainbow colouring for the different curves.
<code>...</code>	Other items passed to function <code>specaccum</code> or <code>plot.specaccum</code> .

Details

These functions provide some alternative methods of obtaining species accumulation results, although function `specaccum` is called by these functions to calculate the actual species accumulation curve.

Functions `accumresult` and `accumcomp` allow to calculate species accumulation curves for subsets of the community and environmental data sets. Function `accumresult` calculates the species accumulation curve for the specified level of a selected environmental variable. Method `accumcomp` calculates the species accumulation curve for all levels of a selected environmental variable separately. Both methods allow to scale the horizontal axis by multiples of the average of a selected continuous variable from the environmental dataset (hint: add the abundance of each site to the environmental data frame to scale accumulation results by mean abundance).

Functions `accumcomp` and `accumplot` provide alternative methods of plotting species accumulation curve results, although function `plot.specaccum` is called by these functions. When you choose to add a legend, make sure that you click in the graph on the spot where you want to put the legend.

Value

The functions provide alternative methods of obtaining species accumulation curve results, although results are similar as obtained by functions `specaccum` and `plot.specaccum`.

Author(s)

Roeland Kindt (World Agroforestry Centre)

References

Kindt, R. & Coe, R. (2005) Tree diversity analysis: A manual and software for common statistical methods for ecological and biodiversity studies.

<http://www.worldagroforestry.org/resources/databases/tree-diversity-analysis>

Examples

```
library(vegan)
data(dune.env)
data(dune)
dune.env$site.totals <- apply(dune,1,sum)
Accum.1 <- accumresult(dune, y=dune.env, scale='site.totals', method='exact', conditioned=TRUE)
Accum.1
accumplot(Accum.1)
accumcomp(dune, y=dune.env, factor='Management', method='exact', legend=FALSE, conditioned=TRUE)
## CLICK IN THE GRAPH TO INDICATE WHERE THE LEGEND NEEDS TO BE PLACED FOR
## OPTION WHERE LEGEND=TRUE (DEFAULT).
```

add.spec.scores

Add Species Scores to Unconstrained Ordination Results

Description

Calculates scores (coordinates) to plot species for PCoA or NMS results that do not naturally provide species scores. The function can also rescale PCA results to use the choice of rescaling used in **vegan** for the `rda` function (after calculating PCA results via PCoA with the euclidean distance first).

Usage

```
add.spec.scores(ordi, comm, method="cor.scores", multi=1, Rscale=F, scaling="1")
```

Arguments

<code>ordi</code>	Ordination result as calculated by <code>cmdscale</code> , <code>isoMDS</code> , <code>sammon</code> , <code>postMDS</code> , <code>metaMDS</code> or <code>NMStandom</code> .
<code>comm</code>	Community data frame with sites as rows, species as columns and species abundance as cell values.
<code>method</code>	Method for calculating species scores. Method "cor.scores" calculates the scores by the correlation between site scores and species vectors (via function <code>cor</code>), method "wa.scores" calculates the weighted average scores (via function <code>wascores</code>) and method "pcoa.scores" calculates the scores by weighing the correlation between site scores and species vectors by variance explained by the ordination axes.
<code>multi</code>	Multiplier for the species scores.
<code>Rscale</code>	Use the same scaling method used by vegan for <code>rda</code> .
<code>scaling</code>	Scaling method as used by <code>rda</code> .

Value

The function returns a new ordination result with new information on species scores. For PCoA results, the function calculates eigenvalues (not sums-of-squares as provided in results from function `cmdscale`), the percentage of explained variance per axis and the sum of all eigenvalues. PCA results (obtained by PCoA obtained by function `cmdscale` with the Euclidean distance) can be scaled as in function `rda`, or be left at the original scale.

Author(s)

Roeland Kindt

References

Kindt, R. & Coe, R. (2005) Tree diversity analysis: A manual and software for common statistical methods for ecological and biodiversity studies.

<http://www.worldagroforestry.org/resources/databases/tree-diversity-analysis>

Examples

```
library(vegan)
data(dune)
distmatrix <- vegdist(dune, method='euc')
## Principal coordinates analysis with 19 axes to estimate total variance
Ordination.model1 <- cmdscale(distmatrix, k=19, eig=TRUE, add=FALSE)
Ordination.model1 <- add.spec.scores(Ordination.model1, dune,
  method='pcoa.scores', Rscale=TRUE, scaling=1, multi=1)
Ordination.model1
## Compare Ordination.model1 with:
Ordination.model2 <- rda(dune)
summary(Ordination.model2, scaling=1)
```

balanced.specaccum *Balanced Species Accumulation Curves*

Description

Provides species accumulation results calculated from balanced (equal subsample sizes) subsampling from each stratum. Sites can be accumulated in a randomized way, or alternatively sites belonging to the same stratum can be kept together. Results are in the same format as `specaccum` and can be plotted with `plot.specaccum` (`vegan`).

Usage

```
balanced.specaccum(comm, permutations=100, strata=strata, grouped=TRUE,
  reps=0, scale=NULL)
```

Arguments

<code>comm</code>	Community data frame with sites as rows, species as columns and species abundance as cell values.
<code>permutations</code>	Number of permutations to calculate the species accumulation curve.
<code>strata</code>	Categorical variable used to specify strata.
<code>grouped</code>	Should sites from the same stratum be kept together (TRUE) or not.
<code>reps</code>	Number of subsamples to be taken from each stratum (see details).
<code>scale</code>	Quantitative variable used to scale the sampling effort (see details).

Details

This function provides an alternative method of obtaining species accumulation results as provided by `specaccum` and `accumresult`.

Balanced sampling is achieved by randomly selecting the same number of sites from each stratum. The number of sites selected from each stratum is determined by `reps`. Sites are selected from strata with sample sizes larger or equal than `reps`. In case that `reps` is smaller than 1 (default: 0), then the number of sites selected from each stratum is equal to the smallest sample size of all strata. Sites from the same stratum can be kept together (`grouped=TRUE`) or the order of sites can be randomized (`grouped=FALSE`).

The results can be scaled by the average accumulation of a quantitative variable (default is number of sites), as in `accumresult` (hint: add the abundance of each site to the environmental data frame to scale accumulation results by mean abundance). When sites are not selected from all strata, then the average is calculated only for the strata that provided sites.

Value

The functions provide alternative methods of obtaining species accumulation curve results, although results are similar as obtained by functions `specaccum` and `accumresult`.

Author(s)

Roeland Kindt (World Agroforestry Centre)

References

Kindt, R., Kalinganire, A., Larwanou, M., Belem, M., Dakouo, J.M., Bayala, J. & Kaire, M. (2008) Species accumulation within landuse and tree diameter categories in Burkina Faso, Mali, Niger and Senegal. *Biodiversity and Conservation*. 17: 1883-1905.

Kindt, R. & Coe, R. (2005) Tree diversity analysis: A manual and software for common statistical methods for ecological and biodiversity studies.

<http://www.worldagroforestry.org/resources/databases/tree-diversity-analysis>

Examples

```
library(vegan)
data(dune.env)
data(dune)
# randomly sample 3 quadrats from each stratum of Management
Accum.1 <- balanced.specaccum(dune, strata=dune.env$Management, reps=3)
Accum.1
dune.env$site.totals <- apply(dune,1,sum)
# scale results by number of trees per quadrat
Accum.2 <- balanced.specaccum(dune, strata=dune.env$Management, reps=3, scale=dune.env$site.totals)
Accum.2
```

BCI.env

Barro Colorado Island Quadrat Descriptions

Description

Environmental characteristics and UTM coordinates of a 50 ha sample plot (consisting of 50 1-ha quadrats) from Barro Colorado Island of Panama. Dataset [BCI](#) provides the tree species composition (trees with diameter at breast height equal or larger than 10 cm) of the same plots.

Usage

```
data(BCI.env)
```

Format

A data frame with 50 observations on the following 6 variables.

UTM.EW a numeric vector

UTM.NS a numeric vector

Precipitation a numeric vector

Elevation a numeric vector

Age.cat a factor with levels c1 c2 c3

Geology a factor with levels pT Tb Tbo Tc Tcm Tct Tgo Tl Tlc

Source

<http://www.sciencemag.org/cgi/content/full/295/5555/666/DC1>

References

Pyke CR, Condit R, Aguilar S and Lao S. (2001). Floristic composition across a climatic gradient in a neotropical lowland forest. *Journal of Vegetation Science* 12: 553-566.

Condit, R, Pitman, N, Leigh, E.G., Chave, J., Terborgh, J., Foster, R.B., Nunez, P., Aguilar, S., Valencia, R., Villa, G., Muller-Landau, H.C., Losos, E. & Hubbell, S.P. (2002). Beta-diversity in tropical forest trees. *Science* 295: 666-669.

Kindt, R. & Coe, R. (2005) Tree diversity analysis: A manual and software for common statistical methods for ecological and biodiversity studies.

http://www.worldagroforestry.org/treesandmarkets/tree_diversity_analysis.asp

Examples

```
data(BCI.env)
```

BiodiversityRGUI

GUI for Biodiversity Analysis and Ordination

Description

This function provides a GUI (Graphical User Interface) for some of the functions of **vegan**, some other packages and some new functions to run biodiversity analysis, including species accumulation curves, diversity indices, Renyi profiles, rank-abundance curves, GLMs for analysis of species abundance and presence-absence, distance matrices, Mantel tests, cluster and ordination analysis (including constrained ordination methods such as RDA, CCA, db-RDA and CAP). The function depends and builds on **Rcmdr**, performing all analyses on the community and environmental datasets that the user selects. A thorough description of the package and the biodiversity and ecological methods that it accomodates (including examples) is provided in the freely available Tree Diversity Analysis manual (Kindt and Coe, 2005).

Usage

```
BiodiversityRGUI()
```

Details

The function launches the R-Commander GUI with an extra menu list for common statistical methods for biodiversity and community ecology analysis.

The R-Commander is launched by changing the location of the Rcmdr "etc" folder to the "etc" folder of BiodiversityR. As the files of the "etc" folder of BiodiversityR are copied from Rcmdr 1.3-14, it is possible that newer versions of the R-Commander will not be launched properly. In such situations, it is possible that copying all files from the Rcmdr "etc" folder again and adding the BiodiversityR menu options to the Rcmdr-menus.txt is all that is needed to launch the R-Commander again.

BiodiversityR uses two data sets for analysis: the community dataset (or community matrix or species matrix) and the environmental dataset (or environmental matrix). The environmental dataset is the same dataset that is used as the "active dataset" of The R-Commander. (Note that you could

sometimes use the same dataset as both the community and environmental dataset. For example, you could use the community dataset as environmental dataset as well to add information about specific species to ordination diagrams. As another example, you could use the environmental dataset as community dataset if you first calculated species richness of each site, saved this information in the environmental dataset, and then use species richness as response variable in a regression analysis.) Some options of analysis of ecological distance allow the community matrix to be a distance matrix (the community data set will be interpreted as distance matrix via `as.dist` prior to further analysis).

BiodiversityR provides the following menu options (each described below in greater detail):

- **Select community dataset** (Community matrix menu) Selects a dataset to be the community dataset.
- **Import datasets from Excel** (Community matrix menu) Imports a community and environmental dataset from an Excel workbook (only applies to a Windows OS).
- **Import datasets from Access** (Community matrix menu) Imports a community and environmental dataset from an Access database (only applies to a Windows OS).
- **View community data set** (Community matrix menu) Invoke the R text editor to view the data of the community data set.
- **Edit community data set** (Community matrix menu) Invoke the R text editor to edit the data of the community data set.
- **Check data sets** (Community matrix menu) Check whether the community and environmental data sets have compatible dimensions.
- **Same sites for community and environmental** (Community matrix menu) Creates a new community dataset with the same sites sequence as the environmental matrix.
- **Make community dataset** (Community matrix menu) Creates a community dataset from the environmental dataset.
- **Remove NA** (Community matrix menu) Removes the same sites with NA from the environmental and community datasets.
- **Transform community matrix** (Community matrix menu) Transforms the community matrix.
- **Select environmental data set** (Environmental matrix menu) Selects a dataset to be the environmental dataset.
- **View environmental data set** (Environmental matrix menu) Invoke the R text editor to view the data of the environmental dataset.
- **Edit environmental data set** (Environmental matrix menu) Invoke the R text editor to edit the data of the environmental dataset.
- **Summary** (Environmental matrix menu) Explores variables of the environmental dataset.
- **Box Cox transformation** (Environmental matrix menu) Creates a transformed variable from one of the variables of the environmental dataset.
- **Species accumulation curves** (Analysis of diversity menu) Estimates and plots species accumulation curves.
- **Diversity indices** (Analysis of diversity menu) Calculates and plots diversity indices.
- **Rank abundance** (Analysis of diversity menu) Calculates and plots rank-abundance curves.

- **Renyi profile** (Analysis of diversity menu) Calculates and plots Renyi diversity profiles.
- **Species abundance as response** (Analysis of species as response menu) Fits and plots regression models assuming that the response variable is count data.
- **Species presence-absence as response** (Analysis of species as response menu) Fits and plots regression models transforming and analysing the response variable as presence-absence.
- **Calculate distance matrix** (Analysis of ecological distance menu) Calculates a distance matrix.
- **Unconstrained ordination** (Analysis of ecological distance menu) Fits and plots unconstrained ordination models.
- **Constrained ordination** (Analysis of ecological distance menu) Fits and plots constrained ordination models.
- **Clustering** (Analysis of ecological distance menu) Calculates and plots results from clustering algorithms.
- **Compare distance matrices** (Analysis of ecological distance menu) Conducts some analysis such as Mantel, MRPP and ANOSIM tests on distance matrices.
- **Help about BiodiversityR** (Help menu) Opens the help file available for the BiodiversityR package (including this html file).
- **Citations for loaded packages** (Help menu) Provides a list of all the loaded packages and gives citation information.
- **Go to website for BiodiversityR** (Help menu) Links to the website for the BiodiversityR package and Tree Diversity Analysis manual.
- **Tree diversity analysis manual** (Help menu) Links to the PDF version of the Tree Diversity Analysis manual. Separate chapters can be downloaded from the website of BiodiversityR (see directly above).

Value

None

Select Community Dataset

This window selects the community dataset to be used in the biodiversity analyses and provides the following options:

- **Data Sets (pick one)** A drop-down list is provided with all the datasets that are available. The current community data set is indicated, or the first data set of the list is shown. New datasets can be loaded through the Data menu of the Rcmdr or through the "import from Excel" option of BiodiversityR (only Windows OS).
- **OK** Make the selected data set the community data set.
- **Cancel** Close the window and do not select a new data set.

Same sites for community and environmental datasets

This window maps the community dataset onto the rownames of the environmental dataset by function `same.sites`. Having the same sequence of sites is an assumption for analysis with BiodiversityR. It may be useful to use this function after making a community dataset from a stacked environmental dataset (especially as sites are ordered in an alphabetic way from the stacked dataset, which may create problems with X1, X10, X100 site names versus the X001, X010 and X100 formats; the function is also useful where some sites do not contain any species). The menu provides the following options:

- **save original community matrix** If this option is selected, the original data set is saved under the name of the community dataset followed by ".orig".
- **OK** Order the sites of the community dataset in exactly the same way as the sites of the environmental data set, leaving out sites that do not have matching names in the environmental data set.
- **Cancel** Close the window and do not re-order and select the sites.

Make Community Dataset

This window selects the variables that indicates sites, species and abundance to create a new community dataset. This dataset becomes the active community dataset. The menu provides the following options:

- **Save result as** The name for the new community dataset.
- **Site variable (rows)** The list shows the variables that can be used for the names of sites (shown as names for the rows). Passed as argument for "row" of function `makecommunitydataset`.
- **Species variable (columns)** The list shows the variables that can be used for the names of species (shown as names for the columns). Passed as argument for "column" of function `makecommunitydataset`.
- **Abundance variable** The list shows the variables that can be used for the abundance values (shown as totals for cells). Passed as argument for "value" of function `makecommunitydataset`.
- **Subset options** The list shows the variables that can be used for the abundance values (shown as totals for cells). Passed as argument for "factor" of function `makecommunitydataset`.
- **Subset** Chooses the value for the subset variable to create the subset. Passed as argument for "level" of function `makecommunitydataset`.
- **OK** Create the community data set and make it the active community dataset.
- **Cancel** Close the window and do not create a new community dataset.

Remove NA

This window removes the sites that have NA (missing values) for a selected variable of the environmental dataset. When environmental variables have missing values, this often creates problems with biodiversity analysis. The menu provides the following options:

- **Select variable** The list shows the variables that can be used to remove sites with NA. Passed as argument for var for functions `removeNAcomm` and `removeNAenv`.
- **OK** Remove the sites with NA.
- **Cancel** Close the window and do not remove the sites with NA.

Transform community matrix

This window transforms the community matrix. The menu provides the following options:

- **Method** Method of transforming the community dataset. Passed as argument for "method" for function `disttransform`. The transformed community matrix is saved under the same name of the original dataset, and the current community dataset therefore becomes the transformed community dataset.
- **Save original community matrix** This option saves the untransformed community dataset by adding `.orig` to the name of the community dataset, as the function replaces the original dataset with the transformed community dataset.
- **OK** Calculate the new community matrix.
- **Cancel** Close the window and do not calculate a new community matrix.

Select Environmental Dataset

This window selects the environmental dataset to be used in the biodiversity analyses. The environmental dataset is always the active dataset for non-Biodiversity Rcmdr options. By selecting the community dataset as the environmental dataset as well, you can also manipulate the community dataset with the other Rcmdr options. The menu provides the following options:

- **Data Sets (pick one)** A drop-down list is provided with all the datasets that are available. The current community data set is indicated, or the first data set of the list is shown. New datasets can be loaded through the Data menu of the Rcmdr or through the "import from Excel" option of BiodiversityR (only in Windows OS).
- **OK** Make the selected data set the environmental data set.
- **Cancel** Close the window and do not select a new data set.

Summary

This window makes a summary of all or a selection of the variables of the environmental dataset, or plots the variables. In case that you want to make a summary of the community dataset, then you need to make the community dataset the environmental dataset at the same time. The menu provides the following options:

- **Select variable** A drop-down list is provided with all the variables of the environmental dataset. The first item of the list (all) is reserved to make a summary of all variables. datasets that are available.
- **OK** Make a summary of all variables or the selected variable by function `summary`.
- **Plot** Plots all variables against each other with function `pairs`, plots a selected continuous variable with function `plot` or plots a categorical with function `boxplot`.
- **Cancel** Close the window and do provide any summary or plot.

Box Cox transformation

This window makes a Box-Cox transformation of a selected variable from the environmental dataset. The menu provides the following options:

- **Select variable** A drop-down list is provided with all the variables of the environmental dataset. Click on the variable to transform.
- **OK** Calculates a Box-Cox transformation of the selected variable with function `box.cox.powers`. Makes a QQ-plot (function `qq.plot`), and performs a Shapiro test (function `shapiro.test`) and Kolmogorov-Smirnov test (function `ks.test`) of the original and transformed variable.
- **Cancel** Close the window.

Species accumulation curves

This window fits and plots species accumulation curves. The menu provides the following options:

- **Save result as** The name for the new object that will save the results from the estimated species accumulation curve after "OK" was clicked, or the name of the object that will be plotted when "Plot" is clicked. In case that you saved a result earlier, then you plot the result by typing in the name of previous result first in this box.
- **Accumulation method** Select the method of species accumulation. Passed as argument for "method" of functions `accumresult` or `accumcomp`.
- **permutations** Number of permutations for random species accumulation. Passed as argument for "permutation" of functions `accumresult` or `accumcomp`.
- **scale of x axis** Method of scaling the horizontal axis. Passed as argument for "scale" of functions `accumresult` or `accumcomp`.
- **subset options** The list shows the variables that can be used for selecting subsets. Option "all" indicates that no subset will be calculated. In case a variable is selected, it will be passed as argument for "factor" of functions `accumresult` or `accumcomp`.
- **Subset** Subset chooses which subsets are calculated. In case that the value of "." (a period) is selected then function `accumcomp` will be used to calculate the species accumulation curve and to plot the curve (you may need to click in the graph to show where the legend needs to be placed). In case another value is chosen, then this will be the argument for "level" of function `accumresult`.
- **Plot options** Options for plotting passed to function `accumplot`.
 - Option "addplot" sets "addit=T" meaning that the species accumulation curve will be added to an existing graph.
 - Option "x limits" sets "xlim". Providing "1,10" will plot between 1 and 10.
 - Option "y limits" sets "ylim". Providing "2,20" will plot between 2 and 20.
 - Option "ci" sets "ci".
 - Option "symbol" sets "pch".
 - Option "cex" sets "cex".
 - Option "colour" sets "col".
- **OK** Calculate the species accumulation curve with functions `accumresult` or `accumcomp`.
- **Plot** Plot the species accumulation curve with the name listed on top with function `accumplot`. You may need to click in the graph to indicate where the legend needs to be placed.
- **Cancel** Close the window and do not calculate a new species accumulation curve.

Diversity indices

The window calculates and fits diversity indices from the community dataset. The menu provides the following options:

- **Save result as** The name for the new object that will save the results from the estimated diversity indices after "OK" was clicked, or the name of the object that will be plotted when "Plot" is clicked. In case that you saved a result earlier, then you plot the result by typing in the name of previous result first in this box. To obtain a meaningful graph, you need to provide similar selections as for the original result (and it may thus be easier to recalculate first and then plot immediately).
- **Diversity index** Select the diversity index. Passed as argument for "index" of functions `diversityresult` or `diversitycomp`.
- **Calculation method** Select the method of calculation. Passed as argument for "method" of functions `diversityresult` or `diversitycomp`.
- **subset options** The list shows the variables that can be used for selecting subsets. Option "all" indicates that no subset will be calculated. In case a variable is selected, it will be passed as argument for "factor" of functions `diversityresult` or `diversitycomp`.
- **Subset** Subset chooses which subsets are calculated. In case that the value of "." (a period) is selected then function `diversitycomp` will be used to calculate the species accumulation curve and to plot the curve (you may need to click in the graph to show where the legend needs to be placed). In case another value is chosen, then this will be the argument for "level" of function `diversityresult`.
- **Output options** Options for obtaining results with functions `diversityresult`, `diversitycomp` or for plotting results.
 - Option "save results" results in adding a new variable with the diversity indices to the environmental dataset. This method only works for calculation method "separate per site" and function `diversityresult`.
 - Option "sort results" results in setting option "sortit=T" for functions `diversityresult` or `diversitycomp`.
 - Option "label results" results in labeling points in the resulting graph.
 - Option "add plot" results in adding points to an existing graph.
 - Option "y limits" results in setting limits for the y axis. Providing "0,10" results in limits of 0 and 10 for the vertical axis.
 - Option "symbol" sets "pch" to choose symbols as in function `points`.
- **OK** Calculate the diversity indices with `diversityresult` or `diversitycomp`.
- **Plot** Plot the diversity results with the name listed on top (should have been calculated first). This will only provide meaningful results if similar options are provided as when calculating the results.
- **Cancel** Close the window and do not calculate new diversity indices.

Rank Abundance

The window fits and plots rank abundance curves for the community dataset. The menu provides the following options:

- **Save result as** The name for the new object that will save the results from the estimated rank abundance curve after "OK" was clicked, or the name of the object that will be plotted when "Plot" is clicked. In case that you saved a result earlier, then you plot the result by typing in the name of previous result first in this box.
- **subset options** The list shows the variables that can be used for selecting subsets. Option "all" indicates that no subset will be calculated. In case a variable is selected, it will be passed as argument for "factor" of functions [rankabundance](#) or [rankabuncomp](#).
- **Subset** Subset chooses which subsets are calculated. In case that the value of "." (a period) is selected then function [rankabuncomp](#) will be used to calculate and plot the rank abundance curves (you may need to click in the graph to show where the legend needs to be placed). In case another value is chosen, then this will be the argument for "level" of function [rankabundance](#).
- **Plot options** The list provides options for scaling the vertical axis. The selection is passed as argument for "scale" of function [rankabunplot](#).
Option "fit RAD" fits distribution models to the observed rank-abundance distribution with function [radfitresult](#) and plots the results.
Option "add plot" sets `addit=T` for function [rankabunplot](#) meaning that the rank abundance curve will be added to an existing graph.
Option "x limits" sets `xlim` for function [rankabunplot](#). Providing "1,10" will plot between 1 and 10.
Option "y limits" sets `ylim` for function [rankabunplot](#). Providing "2,20" will plot between 2 and 20.
- **OK** Calculate the rank abundance curve with functions [rankabundance](#) or [rankabuncomp](#).
- **Plot** Plot the rank abundance curve with the name listed on top (should have been calculated first) with function [rankabunplot](#), or fit models to rank abundance distribution.
- **Cancel** Close the window and do not calculate a new rank abundance curve.

Renyi diversity profiles

The window fits and plots Renyi diversity profiles from the community dataset. The menu provides the following options:

- **Save result as** The name for the new object that will save the results from the diversity profiles after "OK" was clicked, or the name of the object that will be plotted when "Plot" is clicked. In case that you saved a result earlier, then you plot the result by typing in the name of previous result first in this box.
- **Calculation method** The list allows to select the method of calculating the diversity profile. Options "all" and "separate per site" are passed as argument for "method" of function [renyireresult](#). Option "accumulation" results in using function [renyiaccumresult](#). These options are not valid when [renyicomp](#) is invoked (see Subset options).
- **Scale parameters** The "scale parameters" are passed as argument for "scale" for functions [renyireresult](#), [renyiaccumresult](#) or [renyicomp](#).
- **Permutations** The "permutations" are passed as argument for "permutations" for functions [renyiaccumresult](#) or [renyicomp](#).

- **subset options** The list shows the variables that can be used for selecting subsets. Option "all" indicates that no subset will be calculated.
In case a variable is selected, it will be passed as argument for "factor" of functions [renyirestult](#) or [renyicomp](#).
- **Subset** Subset chooses which subsets are calculated. In case that the value of "." (a period) is selected then function [renyicomp](#) will be used to calculate the diversity profile and to plot the curve (you may need to click in the graph to show where the legend needs to be placed). In case another value is chosen, then this will be the argument for "level" of function [renyirestult](#).
- **Plot options** Options for plotting passed to function [renyiplot](#).
Option "evenness profile" sets "evenness=T".
Option "evenness profile" sets addit=T meaning that the diversity profiles will be added to an existing graph.
Option "y limits" sets ylim. Providing "2,20" will plot between 2 and 20.
Option "symbol" sets pch.
Option "cex" sets cex.
Option "colour" sets col.
- **OK** Calculate the diversity profile with functions [renyirestult](#), [renyiaccumresult](#) or [renyicomp](#).
- **Plot** Plot the species accumulation curve with the name listed on top with functions [renyiplot](#) or [persp.renyiaccum](#). The calculation method will determine which plot function is used.
- **Cancel** Close the window and do not calculate a new diversity profile.

Species abundance as response

The window fits and plots regression models for abundance data with a response variable selected from the community dataset and explanatory variables selected from the environmental dataset. (Hint: to analysis species richness patterns, save site-specific species richness (from diversity indices menu) into the environmental data set, and then make the environmental data set to be the community dataset as well). The menu provides the following options:

- **Save result as** The name for the new object that will save the results from the fitted regression model after "OK" was clicked, or the name of the object that will be plotted when "Plot" is clicked. In case that you saved a result earlier, then you can plot the result by typing in the name of previous result first in this box.
- **Model options** Select the method of regression analysis.
Option "linear model" fits a simple linear regression model with function [lm](#).
Option "Poisson model" fits GLMs with Poisson variance functions and log link functions through function [glm](#).
Option "quasi-Poisson model" fits GLMs with quasi-Poisson variance functions and log link functions through function [glm](#).
Option "negative binomial model" fits GLMs with negative binomial variance functions and log link functions through function [glm.nb](#).
Option "gam model" fits GAMs with Poisson variance functions and log link functions through function [gam](#).
Option "gam negbinom model" fits GAMs with negative binomial variance functions and log link functions through function [gam](#).

Option "glmmPQL" fits GLMMs with negative binomial variance functions and log link functions through function `glmmPQL`.

Option "rpart" fits a regression tree through function `rpart`.

- **Standardize** Fit the regression to a standardised dataset with function `scale` (only continuous variables are standardised, not categorical variables).
- **Print summary** Provide a summary of the regression with functions `summary.lm`, `summary.glm` or `summary.gam`.
- **Print anova** Provide a summary of the regression with functions `anova.lm`, `anova.glm`, `anova.gam`, `drop1` or `Anova` (latter two type-II ANOVAs only invoked for multiple regression).
- **add predictions to data frame** Adds the predicted values to the environmental dataset using the model name combined with ".fit" (using the appropriate `predict` function).
- **Response variable** Type the name of the response variable, or select and double-click from the list that is provided. This variable will be displayed on the left-hand side of the formula (variable ~) and is also the response variable that is plotted in the various result plots. The variable is selected as one of the variables (species) of the community dataset, and is first added to the environmental dataset. When you select the environmental dataset to be the community dataset as well, then you can select variables of the environmental dataset as response variable.
- **Explanatory** Type the right-hand side of the model formula (~ explanatory), or select and double-click for variables and select and click for operators to construct the right-hand side of the model formula.
- **Remove site with name** The name of the site to be removed from the environmental dataset.
- **Plot options** The options provide various functions that can be used to plot regression results of the current model (shown on top of the window; should have been estimated first).
 - Option "diagnostic plots" chooses functions `plot.lm` or `gam.check` to plot diagnostic plots. For regression trees, the residuals are plotted against the residuals via `predict.rpart` and `residuals.rpart`.
 - Option "levene test" chooses function `levene.test` and plots residuals of the selected categorical variable (shown on the right).
 - Option "term plot" chooses functions `termplot` or `plot.gam` to plot a termplot of the selected categorical variable (shown on the right).
 - Option "effect plot" chooses function `effect` to plot an effect plot of the selected variable (shown on the right). (The menu option of the R-Commander of models > Graphs plots all the variables).
 - Option "qq plot" chooses function `qq.plot` to plot the residuals from the model.
 - Option "result plot (new)" chooses an appropriate `predict` function to plot a new plot of the model predictions for the selected variable (shown on the right).
 - Option "result plot (add)" chooses an appropriate `predict` function to add a new plot of the model predictions for the selected variable (shown on the right).
 - Option "result plot (interpolate)" chooses an appropriate `predict` function to add a new plot of the model predictions for the selected variable (shown on the right). This model is predicted from a new dataset that only contains 1000 interpolated values for the selected explanatory variable.
 - Option "cr plot" chooses function `cr.plots` to plot a component + residual plots of the selected variable (shown on the right). (The menu option of the R-Commander of models > Graphs plots all the variables).

Option "av plot" chooses function `av.plots` to plot added variable plots of the selected variable (shown on the right). (The menu option of the R-Commander of models > Graphs plots all the variables and has an option of identifying sites with the mouse.)

Option "influence plot" chooses function `influence.plot` to plot influence plots. (The menu option of the R-Commander of models > Graphs includes the option of identifying sites with the mouse.)

Option "multcomp" chooses function `glht` to plot simultaneous confidence intervals of the selected categorical variable (shown on the right).

Option "rpart" chooses functions `plot.rpart` and `text.rpart` to plot a dendrogram for the regression tree result.

- **Plot variable** Variable of the environmental dataset that is used for some plotting functions.
- **OK** Fit the selected models.
- **Plot** Plot results for the model with name that appears on top. The model options need to apply to the model (e.g. if a GLM method was used to fit the model, this option should also be selected when plotting the results).
- **Cancel** Close the window and do not estimate new regression models.

Species presence-absence as response

The window fits and plots regression models for presence-absence data with a response variable selected from the community dataset and explanatory variables selected from the environmental dataset. The menu provides the following options:

- **Save result as** The name for the new object that will save the results from the fitted regression model after "OK" was clicked, or the name of the object that will be plotted when "Plot" is clicked. In case that you saved a result earlier, then you can plot the result by typing in the name of previous result first in this box.
- **Model options** Select the method of regression analysis.
 - Option "crosstab" calculates a cross-tabulation of the selected response (rescaled as presence-absence) and one selected environmental variable, and estimates a Chi-square test of the contingency table with function `chisq.test`.
 - Option "binomial model" fits GLMs with binomial variance functions and logit link functions through function `glm`.
 - Option "quasi-binomial model" fits GLMs with quasi-binomial variance functions and log link functions through function `glm`.
 - Option "gam model" fits GAMs with binomial variance functions and logit link functions through function `gam`.
 - Option "gam quasi-binomial model" fits GAMs with quasi-binomial variance functions and logit link functions through function `gam`.
 - Option "rpart" fits a regression tree through function `rpart`.
 - Option "nnet" fits a forward-feeding artificial neural network through function `nnetrandom`.
- **Standardize** Fit the regression to a standardised dataset with function `scale` (only continuous variables are standardised, not categorical variables).
- **Print summary** Provide a summary of the regression with functions `summary.glm` or `summary.gam`, or use `summary.rpart` or `summary.nnet`

- **Print anova** Provide a summary of the regression with functions `anova.glm`, `anova.gam`, `drop1` or `Anova` (latter two type-II ANOVAs only invoked for multiple regression).
- **add predictions to data frame** Adds the predicted values to the environmental dataset using the model name combined with ".fit" (using the appropriate `predict` function).
- **Response variable** Type the name of the response variable, or select and double-click from the list that is provided. This variable will be displayed on the left-hand side of the formula (variable >0 ~) and is also the response variable that is plotted in the various result plots. The variable is selected as one of the variables (species) of the community dataset, it will be transformed to presence-absence and is first added to the environmental dataset. When you select the environmental dataset to be the community dataset as well, then you can select variables of the environmental dataset as response variable.
- **Explanatory** Type the right-hand side of the model formula (~ explanatory), or select and double-click for variables and select and click for operators to construct the right-hand side of the model formula.
- **Remove site with name** The name of the site to be removed from the environmental dataset.
- **Plot options** The options provide various functions that can be used to plot regression results of the current model (shown on top of the window; should have been estimated first).
 - Option "tabular" chooses function `plot` to plot presence-absence of the response variable against the selected categorical variable (shown on the right).
 - Option "diagnostic plots" chooses functions `plot.lm` or `gam.check` to plot diagnostic plots. For regression trees and artificial neural networks, the predicted values are plotted against the original presence-absence information.
 - Option "levene test" chooses function `levene.test` and plots residuals of the selected categorical variable (shown on the right).
 - Option "term plot" chooses functions `termplot` or `plot.gam` to plot a termplot of the selected categorical variable (shown on the right).
 - Option "effect plot" chooses function `effect` to plot an effect plot of the selected variable (shown on the right). (The menu option of the R-Commander of models > Graphs plots all the variables).
 - Option "qq plot" chooses function `qq.plot` to plot the residuals from the model.
 - Option "result plot (new)" chooses an appropriate `predict` function to plot a new plot of the model predictions for the selected variable (shown on the right).
 - Option "result plot (add)" chooses an appropriate `predict` function to add a new plot of the model predictions for the selected variable (shown on the right)
 - Option "result plot (interpolate)" chooses an appropriate `predict` function to add a new plot of the model predictions for the selected variable (shown on the right). This model is predicted from a new dataset that only contains 1000 interpolated values for the selected explanatory variable.
 - Option "cr plot" chooses function `cr.plots` to plot a component + residual plots of the selected variable (shown on the right). (The menu option of the R-Commander of models > Graphs plots all the variables.)
 - Option "av plot" chooses function `av.plots` to plot added variable plots of the selected variable (shown on the right). (The menu option of the R-Commander of models > Graphs plots all the variables and has an option of identifying sites with the mouse.)

Option "influence plot" chooses function `influence.plot` to plot influence plots. (The menu option of the R-Commander of models > Graphs has an option of identifying sites with the mouse.)

Option "multcomp" chooses function `glht` to plot simultaneous confidence intervals of the selected categorical variable (shown on the right).

Option "rpart" chooses functions `plot.rpart` and `text.rpart` to plot a dendrogram for the regression tree result.

- **Plot variable** Variable of the environmental dataset that is used for some plotting functions.
- **OK** Fit the selected models.
- **Plot** Plot results for the model with name that appears on top. The model options need to apply to the model (e.g. if a GLM method was used to fit the model, this option should also be selected when plotting the results).
- **Cancel** Close the window and do not estimate new regression models.

Calculate distance matrix

This window calculates a distance matrix from the community dataset and provides the following options:

- **Save result as** The name for the new distance matrix that will be calculated after "OK" was clicked.
- **Distance** Ecological distance measure. Passed as argument for "method" for function `vegdist`.
- **Make community dataset** Make the data frame derived from the new distance matrix the active community data set. This distance matrix can be used directly in the other menus for analysis of ecological distance after selecting the "as.dist" options of these windows.
- **OK** Calculate the distance matrix.
- **Cancel** Close the window and do not calculate a new distance matrix.

Unconstrained ordination

The window fits and plots unconstrained ordination models. The menu provides the following options:

- **Save result as** The name for the new object that will save the results from the unconstrained ordination model after "OK" was clicked, or the name of the object that will be plotted when Plot is clicked. In case that you saved a result earlier, then you can plot the result by typing in the name of previous result first in this box.
- **Ordination method** Select the method of ordination analysis.
 - Option "PCA" fits a Principal Components Analysis model with function `rda`.
 - Option "PCA (prcomp)" fits a Principal Components Analysis model with function `prcomp`.
 - Option "PCoA" fits a Principal Coordinates Analysis model with function `cmdscale` using the distance measure selected on the right-hand side (except if the community matrix is interpreted as distance matrix).
 - Option "PCoA (Callez)" fits a Principal Coordinates Analysis model with function `cmdscale` using the distance measure selected on the right-hand side (except if the community matrix is interpreted as distance matrix) and setting `add=T`.

Option "CA" fits a Correspondence Analysis (Reciprocal Averaging) model with function `cca`.
 Option "DCA" fits a Detrended Correspondence Analysis model with function `decorana`.

Option "metaMDS" fits a Non-metric Multidimensional Scaling model with function `metaMDS` using the distance measure selected on the right-hand side (except if the community matrix is interpreted as distance matrix).

Option "NMS (standard)" fits a Non-metric Multidimensional Scaling model with function `NMSrandom` using the distance measure selected on the right-hand side (except if the community matrix is interpreted as distance matrix).

- **Distance** Select the distance measure for the PCoA and NMS methods (other methods have fixed intrinsic distance measures [Euclidean or chi] that can not be changed).

For the methods that provide ordinations based on a distance matrix (PCoA and NMSstandard): passed as argument for "method" for function `vegdist` that calculates the distance matrix first.

Passed as argument for "distance" for function `metaMDS`.

- **PCoA or NMS axes** Select the number of axes to feature in PCoA and NMS results. Passed as argument for "k" for functions `cmdscale`, `metaMDS` or `NMSrandom`.
- **NMS permutation** Select the number of permutations for the NMS results. The solution with the lowest stress after all permutations of random starting positions will be provided. Passed as argument for "trymax" for function `metaMDS` or argument for "perm" for function `NMSrandom`.
- **PCoA or NMS species** Fit species scores to PCoA and NMS results with function `add.spec.scores`. This function adds some other information for PCoA.
- **Model summary** Provide a summary of the ordination with functions `summary.cca`, `summary.decorana` or otherwise list the model object.
- **Scaling** Provide the scaling method. Passed as argument for "scaling" for functions `summary.cca`, `summary.decorana` or `add.spec.scores`.
- **as.dist(Community)** Treat the community dataset as a distance matrix. The community dataset will be used as a distance matrix (via `as.dist`) for unconstrained ordination methods that use a distance matrix as input (`cmdscale` and `NMSrandom` for ordination results and via `ordicluster`, `lines.spantree`, `ordicluster2`, `ordinearest` or `distdisplayed` for plotting options).
- **Plot method** The options provide various functions that can be used to plot ordination results, or to add information to ordination diagrams.

Option "plot" chooses function `plot.cca` to plot results from `rda`, `cca`, `metaMDS` or `decorana` and function `plot` to plot the other ordination results (obtained by function `scores`).

Option "ordiplot" chooses function `ordiplot` to plot ordination results.

Option "ordiplot empty" chooses function `ordiplot` to plot ordination results, but sites and species will be invisible.

Option "identify sites" chooses function `identify.ordiplot` to add names of sites to site symbols (circles) created by function `ordiplot`. You can choose where the name is added by left-clicking in the quadrant next to the symbol where you want to symbol to be plotted. You can stop identifying sites by right-clicking.

Option "identify species" chooses function `identify.ordiplot` to add names of species to species symbols (crosses) created by function `ordiplot`. You can choose where the name is added by left-clicking in the quadrant next to the symbol where you want to symbol to be plotted. You can stop identifying species by right-clicking.

Option "text sites" chooses function `text.ordiplot` to add names of all sites to ordination diagrams created by function `ordiplot`.

Option "text species" chooses function `text.ordiplot` to add names of all species to ordination diagrams created by function `ordiplot`.

Option "points sites" chooses function `points.ordiplot` to add symbols for all sites to ordination diagrams created by function `ordiplot`.

Option "points species" chooses function `points.ordiplot` to add symbols for all species to ordination diagrams created by function `ordiplot`.

Option "origin axes" adds a horizontal and vertical line through the origin of the ordination graph (the origin is the location with coordinates [0,0]).

Option "envfit" chooses function `envfit` to add information for the variable of the environmental dataset selected on the right-hand side to ordination diagrams created by function `ordiplot`.

Option "ordihull" chooses function `ordihull` to add information for the categorical variable of the environmental dataset selected on the right-hand side to ordination diagrams created by function `ordiplot`.

Option "ordiarrows" chooses function `ordiarrows` to add information for the categorical variable of the environmental dataset selected on the right-hand side to ordination diagrams created by function `ordiplot`.

Option "ordisegments" chooses function `ordisegments` to add information for the categorical variable of the environmental dataset selected on the right-hand side to ordination diagrams created by function `ordiplot`.

Option "ordispider" chooses function `ordispider` to add information for the categorical variable of the environmental dataset selected on the right-hand side to ordination diagrams created by function `ordiplot`.

Option "ordiellipse" chooses function `ordiellipse` to add information for the categorical variable of the environmental dataset selected on the right-hand side to ordination diagrams created by function `ordiplot`.

Option "ordisurf" chooses function `ordisurf` to add information for the continuous variable of the environmental dataset selected on the right-hand side to ordination diagrams created by function `ordiplot`.

Option "ordicluster" chooses function `ordicluster` to add information (with distance measure selected in window above - except if the community matrix is interpreted as distance matrix.) to ordination diagrams created by function `ordiplot`.

Option "ordispantree" chooses function `lines.spantree` to add information (with distance measure selected in window above - except if the community matrix is interpreted as distance matrix) to ordination diagrams created by function `ordiplot`.

Option "ordibubble" chooses function `ordibubble` to add information for the continuous variable of the environmental dataset selected on the right-hand side to ordination diagrams created by function `ordiplot`.

Option "ordisymbol" chooses function `ordisymbol` to add information for the categorical variable of the environmental dataset selected on the right-hand side to ordination diagrams created by function `ordiplot`. Make sure that you click in the graph to show where the legend should be placed!

Option "ordivector" chooses function `ordivector` to add information on the selected species of the community dataset selected on the right-hand side to ordination diagrams created by

function `ordiplot`. You should first make the community dataset the environmental dataset to get the list of species on the right-hand side.

Option "ordivector interpretation" chooses function `ordivector` to add information on the selected species of the community dataset selected on the right-hand side to ordination diagrams created by function `ordiplot`. You should first make the community dataset the environmental dataset to get the list of species on the right-hand side. The function will drop down perpendicular lines from each site to the line connecting the origin and the species position.

Option "ordicluster2" chooses function `ordicluster2` to add information (with distance measure selected in window above - except if the community matrix is interpreted as distance matrix) to ordination diagrams created by function `ordiplot`.

Option "ordinearest" chooses function `ordinearest` to add information (with distance measure selected in window above - except if the community matrix is interpreted as distance matrix) to ordination diagrams created by function `ordiplot`.

Option "ordiequilibriumcircle" chooses function `ordiequilibriumcircle` to plot an equilibrium circle to ordination diagrams created by function `ordiplot` from the Principal Components Analysis fitted by `rda`.

Option "distance displayed" compares the distances between each pair of sites in a distance matrix (with distance measure selected in window above) with distances in ordination diagrams created by function `ordiplot` by means of function `distdisplayed`.

Option "screepLOT.cca" provides a screepLOT for PCA results obtained by function `rda` by means of function `screepLOT.cca`.

Option "stress" provides a stress plot (Shepard diagram) for NMS results obtained by function `metaMDS` by means of function `stressplot`.

Option "coenocline" fits coenoclines for all species to the first ordination axis of ordination diagrams created by function `ordiplot` by means of function `ordicoeno`.

- **Plot variable** Variable of the environmental dataset that is used for some plotting functions. For Plot method "ordivector", make the community dataset the environmental dataset first. Some other plot methods may also work with the community dataset as the environmental dataset as well (e.g. "ordibubble", "ordisurf"). Some methods run into problems when the variable has missing observations: in this case, you may need to repeat the ordination analysis after removing sites with missing observations for the variable with the "remove NA" option of the Community dataset menu list.
- **axes** The position of the axes of the ordination result to be plotted in the ordination diagram ("1,2" selects the first two axes of the ordination result). Passed as argument for "choices" for functions `plot.cca`, `scores` or `ordiplot`.
- **add scores to dataframe** Adds the scores of the sites from the `ordiplot` graph to the environmental dataset using the model name combined with ".ax1" and ".ax2".
- **cex** The size of the characters in the resulting plot when "Plot" is clicked.
- **colour** The colour of the resulting plot when "Plot" is clicked.
- **OK** Fit the selected models.
- **Plot** Plot results for the model with name that appears on top. The model options need to apply to the model (e.g. if `rda` was used to fit the model, this option should also be selected when plotting the results).
- **Cancel** Close the window and do not fit or plot ordination models.

Constrained ordination

The window fits and plots constrained ordination models to the community dataset, using variables of the environmental dataset to constrain the ordination model (direct gradient analysis, canonical ordination analysis). The menu provides the following options:

- **Save result as** The name for the new object that will save the results from the unconstrained ordination model after "OK" was clicked, or the name of the object that will be plotted when "Plot" is clicked. In case that you saved a result earlier, then you can plot the result by typing in the name of previous result first in this box.
- **Ordination method** Select the method of ordination analysis.
 - Option "RDA" fits a Redundancy Analysis model with function `rda`.
 - Option "CCA" fits a Canonical Correspondence Analysis (Reciprocal Averaging) model with function `cca`.
 - Option "capscale" fits a scaled Constrained Analysis of Principal Coordinates (distance-based Redundancy Analysis) with function `capscale` using the distance measure selected on the right-hand side (except if the community matrix is interpreted as distance matrix).
 - Option "CAPdiscrim" fits a Constrained Analysis of Principal Coordinates (based on discriminant analysis) with function `CAPdiscrim` using the distance measure selected on the right-hand side (except if the community matrix is interpreted as distance matrix) and the categorical variable selected as explanatory variable.
 - Option "prc" fits principal response curves with function `prc`. To implement the example provided in the documentation for the `prc` function, you need to include the additional steps of defining `pyrifos.env <- data.frame(dose, week)` and making this data set the environmental data set.
 - Option "multiconstrained (RDA)" provides the first row of all ANOVA results (`anova.cca`) for all possible pairwise combinations of the levels of the first explanatory variable (assumed to be a categorical variable) through function `multiconstrained` with `method="rda"`. (When you change contrast to a particular contrast indicator, you obtain an ordination result that can be analyzed further. For several plotting options, you need to change the community and environmental datasets to "newcommunity" and "newenvdata").
 - Option "multiconstrained (CCA)" provides the first row of all ANOVA results (`anova.cca`) for all possible pairwise combinations of the levels of the first explanatory variable (assumed to be a categorical variable) through function `multiconstrained` with `method="cca"`. (When you change contrast to a particular contrast indicator, you obtain an ordination result that can be analyzed further. For several plotting options, you need to change the community and environmental datasets to "newcommunity" and "newenvdata").
 - Option "multiconstrained (capscale)" provides the first row of all ANOVA results (`anova.cca`) for all possible pairwise combinations of the levels of the first explanatory variable (assumed to be a categorical variable) through function `multiconstrained` with `method="capscale"`. (When you change contrast to a particular contrast indicator, you obtain an ordination result that can be analyzed further. For several plotting options, you need to change the community and environmental datasets to "newcommunity" and "newenvdata").
- **Distance** Select the distance measure for the CAP methods (other methods have fixed intrinsic distance measures [Euclidean or chi] that can not be changed). Passed as argument for "dist" for function `capscale` or `CAPdiscrim`. This argument is ignored by the actual functions if the community dataset is interpreted to be a distance matrix already.

- **Model summary** Provide a summary of the ordination with functions `summary.cca` or `summary.prc`, or otherwise list the model object (`CAPdiscrim`).
- **as.dist(Community)** Treat the community dataset as a distance matrix. The community dataset will be used as a distance matrix (via `as.dist`) for constrained ordination methods that can use a distance matrix as input (`capscale` or `CAPdiscrim` for ordination results and via `ordicluster`, `lines.spantree`, `ordicluster2`, `ordinearest` or `distdisplayed` for plotting options).
- **Scaling** Provide the scaling method. This option is not available for function `CAPdiscrim`. Passed as argument for "scaling" for function `summary.cca` or `summary.prc`.
- **permutations** Select the number of permutations for testing the significance of the constrained ordination by Monte-Carlo randomization tests. The default of "0" means that no permutation test will be done. Passed as argument for "permutations" for functions `permutest.cca`, `CAPdiscrim` or `envfit` (one of the plotting options) or as argument for "step" for function `anova.cca` (which is also called by `multiconstrained`).
- **Explanatory** Type the right-hand side of the model formula (~ explanatory), or select and double-click for variables and select and click for operators to construct the right-hand side of the model formula. It is possible to include conditional variables for partial ordination analysis, except for function `CAPdiscrim` and `prc`. For function `prc`, the explanatory variables should be separated by a comma and indicate the "treatment" and "time" factors.
- **Plot method** The options provide various functions that can be used to plot ordination results, or to add information to ordination diagrams.

Option "plot" chooses function `plot.cca` to plot results from `rda`, `cca` or `capscale`, function `plot.prc` to plot results from `prc` and function `plot` to plot the other ordination results (obtained by function `scores`).

Option "ordiplot" chooses function `ordiplot` to plot ordination results.

Option "ordiplot empty" chooses function `ordiplot` to plot ordination results, but sites and species will be invisible.

Option "identify sites" chooses function `identify.ordiplot` to add names of sites to site symbols (circles) created by function `ordiplot`. You can choose where the name is added by left-clicking in the quadrant next to the symbol where you want to symbol to be plotted. You can stop identifying sites by right-clicking.

Option "identify species" chooses function `identify.ordiplot` to add names of species to species symbols (crosses) created by function `ordiplot`. You can choose where the name is added by left-clicking in the quadrant next to the symbol where you want to symbol to be plotted. You can stop identifying species by right-clicking.

Option "identify centroids" chooses function `identify.ordiplot` to add names of centroids to centroid symbols (X) created by function `ordiplot`. You can choose where the name is added by left-clicking in the quadrant next to the symbol where you want to symbol to be plotted. You can stop identifying species by right-clicking.

Option "text sites" chooses function `text.ordiplot` to add names of all sites to ordination diagrams created by function `ordiplot`.

Option "text species" chooses function `text.ordiplot` to add names of all species to ordination diagrams created by function `ordiplot`.

Option "text centroids" chooses function `text.ordiplot` to add names of all centroids to ordination diagrams created by function `ordiplot`.

Option "points sites" chooses function `points.ordiplot` to add symbols for all sites to ordination diagrams created by function `ordiplot`.

Option "points species" chooses function `points.ordiplot` to add symbols for all species to ordination diagrams created by function `ordiplot`.

Option "points centroids" chooses function `points.ordiplot` to add symbols for all centroids to ordination diagrams created by function `ordiplot`.

Option "origin axes" adds a horizontal and vertical line through the origin of the ordination graph (the origin is the location with coordinates [0,0]).

Option "envfit" chooses function `envfit` to add information for the variable of the environmental dataset selected on the right-hand side to ordination diagrams created by function `ordiplot`.

Option "ordihull" chooses function `ordihull` to add information for the categorical variable of the environmental dataset selected on the right-hand side to ordination diagrams created by function `ordiplot`.

Option "ordiarrows" chooses function `ordiarrows` to add information for the categorical variable of the environmental dataset selected on the right-hand side to ordination diagrams created by function `ordiplot`.

Option "ordisegments" chooses function `ordisegments` to add information for the categorical variable of the environmental dataset selected on the right-hand side to ordination diagrams created by function `ordiplot`.

Option "ordispider" chooses function `ordispider` to add information for the categorical variable of the environmental dataset selected on the right-hand side to ordination diagrams created by function `ordiplot`.

Option "ordiellipse" chooses function `ordiellipse` to add information for the categorical variable of the environmental dataset selected on the right-hand side to ordination diagrams created by function `ordiplot`.

Option "ordisurf" chooses function `ordisurf` to add information for the continuous variable of the environmental dataset selected on the right-hand side to ordination diagrams created by function `ordiplot`.

Option "ordicluster" chooses function `ordicluster` to add information (with distance measure selected in window above - except if the community matrix is interpreted as distance matrix) to ordination diagrams created by function `ordiplot`.

Option "ordispantree" chooses function `lines.spantree` to add information (with distance measure selected in window above - except if the community matrix is interpreted as distance matrix) to ordination diagrams created by function `ordiplot`.

Option "ordibubble" chooses function `ordibubble` to add information for the continuous variable of the environmental dataset selected on the right-hand side to ordination diagrams created by function `ordiplot`.

Option "ordisymbol" chooses function `ordisymbol` to add information for the categorical variable of the environmental dataset selected on the right-hand side to ordination diagrams created by function `ordiplot`. Make sure that you click in the graph to show where the legend should be placed!

Option "ordivector" chooses function `ordivector` to add information on the selected species of the community dataset selected on the right-hand side to ordination diagrams created by function `ordiplot`. You should first make the community dataset the environmental dataset to get the list of species on the right-hand side.

Option "ordivector interpretation" chooses function `ordivector` to add information on the selected species of the community dataset selected on the right-hand side to ordination diagrams created by function `ordiplot`. You should first make the community dataset the environmental dataset to get the list of species on the right-hand side. The function will drop down perpendicular lines from each site to the line connecting the origin and the species position.

Option "ordicluster2" chooses function `ordicluster2` to add information (with distance measure selected in window above - except if the community matrix is interpreted as distance matrix) to ordination diagrams created by function `ordiplot`.

Option "ordinearest" chooses function `ordinearest` to add information (with distance measure selected in window above - except if the community matrix is interpreted as distance matrix) to ordination diagrams created by function `ordiplot`.

Option "distance displayed" compares the distances between each pair of sites in a distance matrix (with distance measure selected in window above) with distances in ordination diagrams created by function `ordiplot` by means of function `distdisplayed`.

Option "coenocline" fits coenoclines for all species to the first ordination axis of ordination diagrams created by function `ordiplot` by means of function `ordicoeno`.

- **Plot variable** Variable of the environmental dataset that is used for some plotting functions. For Plot method "ordivector", make the community dataset the environmental dataset first. Some other plot methods may also work with the community dataset as the environmental dataset as well (e.g. "ordibubble", "ordisurf"). Some methods run into problems when the variable has missing observations: in this case, you may need to repeat the ordination analysis after removing sites with missing observations for the variable with the "remove NA" option of the Community dataset menu list.
- **axes** The position of the axes of the ordination result to be plotted in the ordination diagram ("1,2" selects the first two axes of the ordination result). Passed as argument for "choices" for functions `plot.cca`, `scores` or `ordiplot`.
- **add scores to dataframe** Adds the scores of the sites from the `ordiplot` graph to the environmental dataset using the model name combined with ".ax1" and ".ax2".
- **cex** The size of the characters in the resulting plot when "Plot" is clicked.
- **colour** The colour of the resulting plot when "Plot" is clicked.
- **OK** Fit the selected models.
- **Plot** Plot results for the model with name that appears on top. The model options need to apply to the model (e.g. if `rda` was used to fit the model, this option should also be selected when plotting the results).
- **Cancel** Close the window and do not fit or plot ordination models.

Clustering

This window performs various methods of cluster analysis based on the information of the community dataset. The menu provides the following options:

- **Save cluster as** The name for the new object that will save the results from the cluster analysis after "OK" was clicked, or the name of the object that will be plotted when "Plot" is clicked. In case that you saved a result earlier, then you can plot the result by typing in the name of previous result first in this box.

- **Cluster method** Select the method of ordination analysis.
 - Option "hclust" results in a cluster analysis fitted by function `hclust`. The distance for the distance matrix derived from the community dataset is selected on the right-hand side.
 - Option "agnes" results in a cluster analysis fitted by function `agnes`. The distance for the distance matrix derived from the community dataset is selected on the right-hand side.
 - Option "diana" results in a cluster analysis fitted by function `diana`. The distance for the distance matrix derived from the community dataset is selected on the right-hand side.
 - Option "kmeans" results in a cluster analysis fitted by function `kmeans`. This method is based on the Euclidean distance.
 - Option "cascadeKM" results in a cluster analysis fitted by function `cascadeKM`. This method is based on the Euclidean distance as it is based on K-means clustering.
 - Option "pam" results in a cluster analysis fitted by function `pam`. The distance for the distance matrix derived from the community dataset is selected on the right-hand side.
 - Option "clara" results in a cluster analysis fitted by function `clara`. The distance for the distance matrix derived from the community dataset is selected on the right-hand side.
 - Option "fanny" results in a cluster analysis fitted by function `fanny`. The distance for the distance matrix derived from the community dataset is selected on the right-hand side.
- **Distance** Ecological distance measure used for the distance matrix. Passed as argument for "method" for function `vegdist`.
- **as.dist(Community)** Treat the community dataset as a distance matrix. The community dataset will be used as a distance matrix (via `as.dist`). This option is not available for `kmeans`).
- **cluster summary** Provide the results of the cluster analysis with `summary.agnes`, `summary.diana`, `summary.pam`, `summary.clara` or `summary.fanny` or provide results of `hclust` or `kmeans`
- **cophenetic correlation** Calculate the correlation of the distances in the distance matrix with the distances in the dendrogram (estimated with function `cophenetic`) by the Mantel test (`mantel`). It only works for hierarchical clustering methods (`hclust`, `agnes` and `diana`).
- **clusters** Determine a fixed number of clusters.
 - This number selects the number of clusters to be calculated by the non-hierarchical cluster methods as it is passed as argument for "centers" for function `kmeans` and argument for "k" for functions `pam`, `clara` and `fanny`.
 - This number selects the number of groups for the partition with the largest number of groups of the cascade as it is passed as argument for "sup.gr" for function `cascadeKM` (the argument for "inf.gr" is set to "2").
 - This number selects the number of clusters to be reported for cluster membership for hierarchical clustering methods (`hclust`, `agnes` and `diana`) as determined by function `cutree`: passed as argument for "k" for this function.
 - This number selects the number of rectangles to be plotted on a dendrogram with plotting option of "rectangles": passed as argument for "k" for function `rect.hclust`.
 - This number selects the number of clusters to be plotted with plotting option of "pruned dendrogram": passed as argument for "k" for function `clip.clust`.
- **Save cluster membership** Save the identity of the cluster to which each site belongs into the environmental data set. For hierarchical clustering methods (`hclust`, `agnes` and `diana`) as determined by function `cutree`, with parameter "k" obtained from the box above.

- **Cluster options** Choose the options that are available for some of the hierarchical clustering methods.
Options "average", "single", "complete", "ward", "median" and "centroid" can be passed meaningfully as argument for "method" for `hclust`.
Options "average", "single", "complete", "ward" and "weighted" can be passed meaningfully as argument for "method" for `agnes`.
- **Plot options** Choose the options that are available for plotting hierarchical clustering results (except for "cascadeKM").
Option "dendrogram1" selects function `plot.hclust`, `plot.agnes` or `plot.diana` to plot clustering results.
Option "dendrogram2" selects function `plot.hclust`, `plot.agnes` or `plot.diana` to plot clustering results with argument `hang` set to -1. This option will result in each branch of the dendrogram to reach "ground level".
Option "rectangles" selects function `rect.hclust` to plot rectangles around the number of cluster determined by option "clusters" selected above.
Option "pruned dendrogram" selects function `clip.clust` to prune the cluster to the number of cluster selected by option "clusters" selected above. This option may only work with cluster results obtained by `plot.hclust`.
Option "kgs" selects function `kgs` as one method of selecting the optimal number of clusters and plots its results.
Option "cophenetic" uses function `cophenetic` to the distance in the dendrogram against the distance of the distance matrix (calculated earlier for the clustering algorithm). A reference line ($y=x$) is added to the graph.
Option "cascadeKM" selects function `plot.cascadeKM` to plot results obtained by function `cascadeKM`.
- **cex** The size of the characters in the resulting plot when "Plot" is clicked.
- **colour** The colour of the resulting plot when "Plot" is clicked.
- **OK** Fit the selected models.
- **Plot** Plot results for the cluster with name that appears on top. Plotting will only be meaningful for hierarchical methods (`hclust`, `agnes` and `diana`).
- **Cancel** Close the window and do not analyse or plot clusters..

Compare distance matrices

This window calculates a distance matrix from the community dataset. This distance matrix can be analysed by a Mantel, MRPP or ANOSIM test based on information from the environmental dataset. You can compare two different community datasets if you make one the community dataset and the other one the environmental dataset. The menu provides the following options:

- **Type of test** Selects the type of test to be used.
Option "mantel" results in a Mantel test estimated by function `mantel`. The distance for distance matrix derived from the community dataset is selected below, the distance to be derived from the environmental dataset is selected on the right-hand side.
Option "anosim" results in a ANOSIM test estimated by function `anosim` as summarized by `summary.anosim`. The distance measure for the distance matrix derived from the community

dataset is selected below, the categorical variable of the environmental dataset is selected at the right-hand side.

Option "mrpp" results in a MRPP test estimated by function `mrpp`. The distance measure for the distance matrix derived from the community dataset is selected below, the categorical variable of the environmental dataset is selected at the right-hand side.

Option "rankindex" results in a series of Mantel tests with a series of distance measures selected by function `rankindex` for the community dataset and the Euclidean distance for the environmental dataset (except for datasets that contain factors where `daisy` is used).

- **Environmental distance** The environmental distance is only used for the test option of "mantel" (test option of "rank index" makes its own choice in between "daisy" or "euclidean" distance). The distance determines the type of distance matrix that is obtained from the environmental data set.

Option "daisy" results in function `daisy` to be used for providing the distance matrix. This is the only realistic method for environmental datasets that contain categorical variables.

The other options are passed as arguments for "method" for function `vegdist`.

- **Community distance** Ecological distance measure used for the distance matrix obtained from the community data set.

Passed as argument for "method" for function `vegdist`.

For the "rankindex" type of test, a series of distance measures are tested automatically.

- **Environmental variable** Selection of the environmental variable(s). Some methods run into problems when the variable has missing observations: in this case, you may need to repeat the ordination analysis after removing sites with missing observations for the variable with the "remove NA" option of the Community dataset menu list.

For test option "mantel", when "all" is selected, then the distance matrix is calculated for all variables of the environmental dataset. For environmental datasets with some categorical variables, only environmental distance "daisy" will result in actual distance matrices.

For test option "mantel", when a variable is selected, then the distance matrix is only calculated for that variable. In case that the variable is categorical, then the `daisy` distance is used automatically.

For test option "anosim", the selected environmental variable is passed as argument for "grouping" for function `anosim`.

For test option "mrpp", the selected environmental variable is passed as argument for "grouping" for function `mrpp`.

For test option "rankindex", when "all" is selected, then the environmental dataset is passed as argument for "grad" for function `rankindex`.

For test option "rankindex", the selected variable is passed as argument for "grad" for function `rankindex`.

- **as.dist(Community)** Treat the community dataset as a distance matrix. The community dataset will be used as a distance matrix (via `as.dist`).
- **Plot results** Plots the distances of the community dataset against the distance of the environmental dataset for test options "mantel", "anosim" and "mrpp". For categorical variables (the only possibility for "anosim" and "mrpp"), environmental distance equals "0" if sites belong to the same group and "1" if they belong to a different group except if they are ordered categorical variables (depending on the results of the `daisy` distance; for ordered factors, it is recommended to create a new factor that is unordered and use this variable for the analysis; see `factor`).

- **permutations** Number of permutations. Passed as argument for "permutations" for functions `mantel`, `anosim` and `mrpp`.
- **correlation** Correlation method. Passed as argument for "method" for function `mantel`.
- **OK** Estimate the selected tests.
- **Cancel** Close the window and do estimate a new test.

Author(s)

Roeland Kindt (with some help from Jari Oksanen)

References

Kindt, R. & Coe, R. (2005) Tree diversity analysis: A manual and software for common statistical methods for ecological and biodiversity studies.

<http://www.worldagroforestry.org/resources/databases/tree-diversity-analysis>

CAPdiscrim

Canonical Analysis of Principal Coordinates based on Discriminant Analysis

Description

This function provides a method for CAP as described by the authors of the ordination method. The CAP method implemented in **vegan** through `capscale` conforms more to distance-based Redundancy Analysis (Legendre & Anderson, 1999) than to the original description for CAP (Anderson & Willis, 2003).

Usage

```
CAPdiscrim(formula,data,dist="bray",axes=4,m=0,permutations=0)
```

Arguments

formula	Formula with a community data frame (with sites as rows, species as columns and species abundance as cell values) or distance matrix on the left-hand side and a categorical variable on the right-hand side (only the first explanatory variable will be used).
data	Environmental data set.
dist	Method for calculating ecological distance with function <code>vegdist</code> : partial match to "manhattan", "euclidean", "canberra", "bray", "kulczynski", "jaccard", "gower", "morisita", "horn" or "mountford". This argument is ignored in case that the left-hand side of the formula already is a distance matrix.
axes	Number of PCoA axes (<code>cmdscale</code>) to provide in the result.
m	Number of PCoA axes to be investigated by discriminant analysis (<code>lda</code>). If m=0 then the number of axes that provides the best distinction between the groups is calculated (following the method of Anderson and Willis).
permutations	The number of permutations for significance testing.

Details

This function provides a method of Constrained Analysis of Principal Coordinates (CAP) that conforms to the description of the method by the developers of the method, Anderson and Willis. The method investigates the results of a Principal Coordinates Analysis (function `cmdscale`) with linear discriminant analysis (`lda`). Anderson and Willis advocate to use the number of principal coordinate axes that result in the best prediction of group identities of the sites.

For permutations > 0 , the analysis is repeated by randomising the observations of the environmental data set. The significance is estimated by dividing the number of times the randomisation generated a larger percentage of correct predictions.

Value

The function returns an object with information on CAP based on discriminant analysis. The object contains following elements:

PCoA	the positions of the sites as fitted by PCoA
m	the number of axes analysed by discriminant analysis
tot	the total variance (sum of all eigenvalues of PCoA)
varm	the variance of the m axes that were investigated
group	the original group of the sites
CV	the predicted group for the sites by discriminant analysis
percent	the percentage of correct predictions
x	the positions of the sites provided by the discriminant analysis
F	the squares of the singular values of the discriminant analysis
manova	the results for MANOVA with the same grouping variable
signi	the significance of the percentage of correct predictions
manova	a summary of the observed randomised prediction percentages

The object can be plotted with `ordiplot`, and species scores can be added by `add.spec.scores`.

Author(s)

Roeland Kindt (World Agroforestry Centre)

References

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

Anderson, M.J. & Willis, T.J. (2003). Canonical analysis of principal coordinates: a useful method of constrained ordination for ecology. *Ecology* 84: 511-525.

Kindt, R. & Coe, R. (2005) Tree diversity analysis: A manual and software for common statistical methods for ecological and biodiversity studies.

<http://www.worldagroforestry.org/resources/databases/tree-diversity-analysis>

Examples

```
library(vegan)
library(MASS)
data(dune)
data(dune.env)
Ordination.model1 <- CAPdiscrim(dune~Management, data=dune.env,
  dist="bray", axes=2, m=0)
Ordination.model1
plot1 <- ordiplot(Ordination.model1)
ordisymbol(plot1, dune.env, "Management", legend=FALSE)
## CLICK IN THE GRAPH TO INDICATE THE POSITION FOR THE LEGEND
## IN CASE THAT THE OPTION WAS LEGEND=TRUE.
```

caprescale	<i>Rescaling of Capscale Results to Reflect Total Sums of Squares Of Distance Matrix</i>
------------	--

Description

This is a simple function that rescales the ordination coordinates obtained from the distance-based redundancy analysis method implemented in **vegan** through [capscale](#). The rescaling of the ordination coordinates results in the distances between fitted site scores in ordination results (scaling=1 obtained via [ordiplot](#) to be equal to the distances between sites on the axes corresponding to positive eigenvalues obtained from principal coordinates analysis ([cmdscale](#)).

Usage

```
caprescale(x, verbose=FALSE)
```

Arguments

x	Ordination result obtained with capscale .
verbose	Give some information on the pairwise distances among sites (TRUE) or not.

Details

The first step of distance-based redundancy analysis involves principal coordinates analysis whereby the distances among sites from a distance matrix are approximated by distances among sites in a multidimensional configuration (ordination). In case that the principal coordinates analysis does not result in negative eigenvalues, then the distances from the distance matrix are the same as the distances among the sites in the ordination. In case that the principal coordinates analysis results in negative eigenvalues, then the distances among the sites on all ordination axes are related to the sum of positive eigenvalues, a sum which is larger than the sum of squared distances of the distance matrix.

The distance-based redundancy analysis method implemented in **vegan** through [capscale](#) uses a specific rescaling method for ordination results. Function [caprescale](#) modifies the results of [capscale](#) so that an ordination with scaling=1 (a distance biplot) obtained via [ordiplot](#) preserves

the distances reflected in the principal coordinates analysis implemented as the first step of the analysis. See Legendre and Legendre (1998) about the relationship between fitted site scores and eigenvalues.

Value

The function modifies and returns an object obtained via [capscale](#).

Author(s)

Roeland Kindt (World Agroforestry Centre)

References

- Legendre, P. & Legendre, L. (1998). Numerical Ecology. Amsterdam: Elsevier. 853 pp.
- Legendre, P. & Anderson, M.J. (1999). Distance-based redundancy analysis: testing multispecies responses in multifactorial ecological experiments. Ecological Monographs 69: 1-24.

Examples

```
library(vegan)
library(MASS)
data(dune)
data(dune.env)
Distmatrix.1 <- vegdist(dune,method='bray')
Ordination.model1 <- cmdscale(Distmatrix.1, k=19, eig=TRUE, add=FALSE)
# Sum of all eigenvalues
sum(Ordination.model1$eig)
# [1] 4.395807541512926
sum(Ordination.model1$eig[1:14])
# [1] 4.593946896588808
Distmatrix.2 <- as.matrix(vegdist(Ordination.model1$points[,1:14],method='euc'))
totalsumsquares1 <- sum(Distmatrix.2^2)/(2*20)
# Sum of distances among sites in principal coordinates analysis on axes
# corresponding to positive eigenvalues
totalsumsquares1
# [1] 4.593946896588808
Ordination.model2 <- caprescale(dune ~ Management,dune.env,dist='bray', add=FALSE)
# Total sums of positive eigenvalues of the distance-based redundancy analysis
Ordination.model2$CA$tot.chi+Ordination.model2$CCA$tot.chi
# [1] 4.593946896588808
Ordination.model3 <- caprescale(Ordination.model2, verbose=TRUE)
sum1 <- summary(Ordination.model3,axes=17,scaling=1)$constraints
Distmatrix.3 <- as.matrix(vegdist(sum1 ,method='euc'))
totalsumsquares2 <- sum((Distmatrix.3)^2)/(2*20)/19
totalsumsquares2
# [1] 4.593946896588808
```

crosstabanalysis *Presence-absence Analysis by Cross Tabulation*

Description

This function makes a cross-tabulation of two variables after transforming the first variable to presence-absence and then returns results of `chisq.test`.

Usage

```
crosstabanalysis(x, variable, factor)
```

Arguments

x	Data set that contains the variables "variable" and "factor".
variable	Variable to be transformed in presence-absence in the resulting cross-tabulation.
factor	Variable to be used for the cross-tabulation together with the transformed variable.

Value

The function returns the results of `chisq.test` on a crosstabulation of two variables, after transforming the first variable to presence-absence first.

Author(s)

Roeland Kindt

References

Kindt, R. & Coe, R. (2005) Tree diversity analysis: A manual and software for common statistical methods for ecological and biodiversity studies.

<http://www.worldagroforestry.org/resources/databases/tree-diversity-analysis>

Examples

```
library(vegan)
data(dune.env)
crosstabanalysis(dune.env, "Manure", "Management")
```

deviancepercentage *Calculate Percentage and Significance of Deviance Explained by a GLM*

Description

This function calculates the percentage of deviance explained by a GLM model and calculates the significance of the model.

Usage

```
deviancepercentage(x,data,test="F",digits=2)
```

Arguments

x	Result of GLM as calculated by <code>glm</code> or <code>glm.nb</code> .
data	Data set to be used for the null model (preferably the same data set used by the 'full' model).
test	Test statistic to be used for the comparison between the null model and the 'full' model as estimated by <code>anova.glm</code> or <code>anova.negbin</code> : partial match of one of "Chisq", "F" or "Cp".
digits	Number of digits in the calculation of the percentage.

Details

The function calculates the percentage of explained deviance and the significance of the 'full' model by contrasting it with the null model.

For the null model, the data is subjected to `na.omit`. You should check whether the same data are used for the null and 'full' models.

Value

The function calculates the percentage of explained deviance and the significance of the 'full' model by contrasting it with the null model by ANOVA. The results of the ANOVA are also provided.

Author(s)

Roeland Kindt

References

Kindt, R. & Coe, R. (2005) Tree diversity analysis: A manual and software for common statistical methods for ecological and biodiversity studies.

<http://www.worldagroforestry.org/resources/databases/tree-diversity-analysis>

Examples

```
library(vegan)
data(dune)
data(dune.env)
dune.env$Agrostol <- dune$Agrostol
Count.model1 <- glm(Agrostol ~ Management + A1, family=quasipoisson(link=log),
  data=dune.env, na.action=na.omit)
summary(Count.model1)
deviancepercentage(Count.model1, dune.env, digits=3)
```

 dist.eval

Distance Matrix Evaluation

Description

Function `dist.eval` provides one test of a distance matrix, and then continues with `distconnected` (**vegan**). Function `bioenv.numeric` applies `bioenv` (**vegan**), but allows to convert variables to numeric variables and excludes all categorical variables from the environmental data set.

Usage

```
dist.eval(x,dist)
## Default S3 method:
bioenv.numeric(comm, env, method = "spearman", index = "bray",
  as.numeric = c(),
  upto = ncol(env), trace = FALSE, partial = NULL, ...)
```

Arguments

<code>x, comm</code>	Community data frame with sites as rows, species as columns and species abundance as cell values.
<code>env</code>	Environmental data frame with sites as rows and variables as columns.
<code>dist, index</code>	Method for calculating ecological distance with function <code>vegdist</code> : partial match to "manhattan", "euclidean", "canberra", "bray", "kulczynski", "jaccard", "gower", "morisita", "horn" or "mountford".
<code>method</code>	The correlation method. See also <code>cor</code> and <code>bioenv</code> .
<code>upto</code>	Maximum number of parameters in studied subsets. See also <code>bioenv</code> .
<code>trace</code>	Trace the calculations. See also <code>bioenv</code> .
<code>partial</code>	Dissimilarities partialled out. See also <code>bioenv</code> .
<code>...</code>	Other arguments passed. See also <code>cor</code> and <code>bioenv</code> .
<code>as.numeric</code>	Vector with names of variables in the environmental data set to be converted to numeric variables.

Details

Function `dist.eval` provides two tests of a distance matrix:

(i) The first test checks whether any pair of sites that share some species have a larger distance than any other pair of sites that do not share any species. In case that cases are found, then a warning message is given.

(ii) The second test is the one implemented by the `distconnected` function (**vegan**). The `distconnected` test is only calculated for distances that calculate a value of 1 if sites share no species (i.e. not manhattan or euclidean), using the threshold of 1 as an indication that the sites do not share any species. Interpretation of analysis of distance matrices that provided these warnings should be cautious.

Function `bioenv.numeric` provides some simple methods of dealing with categorical variables prior to applying `bioenv`.

Value

The function tests whether distance matrices have some desirable properties and provide warnings if this is not the case.

Author(s)

Roeland Kindt (World Agroforestry Centre)

References

Kindt, R. & Coe, R. (2005) Tree diversity analysis: A manual and software for common statistical methods for ecological and biodiversity studies.

<http://www.worldagroforestry.org/resources/databases/tree-diversity-analysis>

Examples

```
library(vegan)
data(dune)
dist.eval(dune,"euclidean")
dist.eval(dune,"bray")
```

dist.zeroes

Distance Matrix Transformation

Description

Sample units without any species result in "NaN" values in the distance matrix for some of the methods of `vegdist` (**vegan**). The function replaces "NA" by "0" if both sample units do not contain any species and "NA" by "1" if only one sample unit does not have any species.

Usage

```
dist.zeroes(comm,dist)
```

Arguments

comm	Community data frame with sites as rows, species as columns and species abundance as cell values.
dist	Distance matrix as calculated with function <code>vegdist</code> .

Details

This functions changes a distance matrix by replacing "NaN" values by "0" if both sample units do not contain any species and by "1" if only one sample unit does not contain any species.

Please note that there is a valid reason (deliberate removal of zero abundance values from calculations) that the original distance matrix contains "NaN", so you may not wish to do this transformation and remove sample units with zero abundances from further analysis.

Value

The function provides a new distance matrix where "NaN" values have been replaced by "0" or "1".

Author(s)

Roeland Kindt (World Agroforestry Centre)

References

Kindt, R. & Coe, R. (2005) Tree diversity analysis: A manual and software for common statistical methods for ecological and biodiversity studies.

<http://www.worldagroforestry.org/resources/databases/tree-diversity-analysis>

Examples

```
library(vegan)
matrix <- array(0,dim=c(5,3))
matrix[4,] <- c(1,2,3)
matrix[5,] <- c(1,0,0)
dist1 <- vegdist(matrix,method="kulc")
dist1
dist2 <- dist.zeroes(matrix,dist1)
dist2
```

distdisplayed

Compare Distance Displayed in Ordination Diagram with Distances of Distance Matrix

Description

This function compares the distance among sites as displayed in an ordination diagram (generated by `ordiplot`) with the actual distances among sites as available from a distance matrix (as generated by `vegdist`).

Usage

```
distdisplayed(x, ordiplot, distx = "bray", plotit = T, addit = F,
              method = "spearman", permutations = 100, abline = F, gam = T, ...)
```

Arguments

x	Community data frame (with sites as rows, species as columns and species abundance as cell values) or distance matrix.
ordiplot	Ordination diagram generated by ordiplot or distance matrix.
distx	Ecological distance used to calculate the distance matrix (theoretically the same distance as displayed in the ordination diagram); passed to vegdist and partial match to "manhattan", "euclidean", "canberra", "bray", "kulczynski", "jaccard", "gower", "morisita", "horn", "mountford", "raup", "binomial" or "chao". This argument is ignored in case that "x" is already a distance matrix.
plotit	Should a plot comparing the distance in ordination diagram (or the distance matrix) with the distance from the distance matrix be generated (or not).
addit	Should the GAM regression result be added to an existing plot (or not).
method	Method for calculating the correlation between the ordination distance and the complete distance; from function mantel passed to function cor : "pearson", "spearman" or "kendall".
permutations	Number of permutations to assess the significance of the Mantel test; passed to mantel .
abline	Should a reference line (y=x) be added to the graph (or not).
gam	Evaluate the correspondence between the original distance and the distance from the ordination diagram with GAMs estimated by gam .
...	Other arguments passed to mantel .

Details

This function compares the Euclidean distances (between sites) displayed in an ordination diagram with the distances of a distance matrix. Alternatively, the distances of one distance matrix are compared against the distances of another distance matrix.

These distances are compared by a Mantel test ([mantel](#)) and (optionally) a GAM regression ([gam](#)). Optionally, a graph is provided comparing the distances and adding GAM results. .

Value

The function returns the results of a Mantel test and (optionally) the results of a GAM analysis.

Author(s)

Roeland Kindt (World Agroforestry Centre)

References

Kindt, R. & Coe, R. (2005) Tree diversity analysis: A manual and software for common statistical methods for ecological and biodiversity studies.

<http://www.worldagroforestry.org/resources/databases/tree-diversity-analysis>

Examples

```
library(vegan)
library(mgcv)
data(dune)
distmatrix <- vegdist(dune,method="kulc")
ordination.model1 <- cmdscale(distmatrix,k=2)
ordiplot1 <- ordiplot(ordination.model1)
distdisplayed(dune,ordiplot=ordiplot1,distx="kulc",plotit=TRUE,
  method="spearman",permutations=100,gam=TRUE)
```

disttransform

Community Matrix Transformation

Description

Transforms a community matrix. Some transformation methods are described by distances for the original community matrix that result in the same distance matrix as calculated with the euclidean distance from the transformed community matrix. In several cases (methods of "hellinger", "chord", "profiles" and "chi.square"), the method makes use of function `decostand`. In several other cases ("Braun.Blanquet", "Domin", "Hult", "Hill", "fix" and "coverscale.log"), the method makes use of function `coverscale`.

Usage

```
disttransform(x, method="hellinger")
```

Arguments

x	Community data frame with sites as rows, species as columns and species abundance as cell values.
method	Distance measure for the original community matrix that the euclidean distance will calculate for the transformed community matrix: partial match to "hellinger", "chord", "profiles", "chi.square", "log", "square", "pa", "Braun.Blanquet", "Domin", "Hult", "Hill", "fix" and "coverscale.log".

Details

This functions transforms a community matrix.

Some transformation methods ("hellinger", "chord", "profiles" and "chi.square") have the behaviour that the euclidean distance from the transformed matrix will equal a distance of choice for the original matrix. For example, using method "hellinger" and calculating the euclidean distance will result

in the same distance matrix as by calculating the Hellinger distance from the original community matrix.

Transformation methods ("Braun.Blanquet", "Domin", "Hult", "Hill", "fix" and "coverscale.log") call function `coverscale`.

Value

The function returns a transformed community matrix.

Author(s)

Roeland Kindt (World Agroforestry Centre)

References

Legendre, P. & Gallagher, E.D. (2001). Ecologically meaningful transformations for ordination of species data. *Oecologia* 129: 271-280.

Kindt, R. & Coe, R. (2005) Tree diversity analysis: A manual and software for common statistical methods for ecological and biodiversity studies.

<http://www.worldagroforestry.org/resources/databases/tree-diversity-analysis>

Examples

```
library(vegan)
data(dune)
Community.1 <- disttransform(dune, method='hellinger')
Distmatrix.1 <- vegdist(Community.1,method='euclidean')
Distmatrix.1
```

diversityresult

Alternative Diversity Results

Description

Provides alternative methods of obtaining results on diversity statistics than provided directly by functions `diversity`, `fisher.alpha`, `specpool` and `specnumber` (all from **vegan**), although these same functions are called. Some other statistics are also calculated such as the reciprocal Berger-Parker diversity index and abundance (not a diversity statistic). The statistics can be calculated for the entire community, for each site separately, the mean of the sites can be calculated or a jackknife estimate can be calculated for the community.

Usage

```
diversityresult(x,y="", factor, level, index="Shannon",method="all",sortit=F,
  digits=8)
diversitycomp(x,y="", factor1,factor2="", index="Shannon",
  method="all",sortit=F,...)
```

Arguments

x	Community data frame with sites as rows, species as columns and species abundance as cell values.
y	Environmental data frame.
factor	Variable of the environmental data frame that defines subsets to calculate diversity statistics for.
level	Level of the variable to create the subset to calculate diversity statistics.
index	Type of diversity statistic with "richness" to calculate species richness, "abundance" to calculate abundance, "Shannon" to calculate the Shannon diversity index, "Simpson" to calculate 1-Simpson concentration index, "inverseSimpson" to calculate the reciprocal Simpson diversity index, "Logalpha" to calculate the log series alpha diversity index, "Berger" to calculate the reciprocal Berger-Parker diversity index, "Jevenness" to calculate one Shannon evenness index, "Evenness" to calculate another Shannon evenness index, "jack1" to calculate the first-order jackknife gamma diversity estimator, "jack2" to calculate the second-order jackknife gamma diversity estimator, "chao" to calculate the Chao gamma diversity estimator and "boot" to calculate the bootstrap gamma diversity estimator.
method	Method of calculating the diversity statistics: "all" calculates the diversity of the entire community (all sites pooled), "s" calculates the diversity of each site separately, "mean" calculates the average diversity of the sites, "sd" calculates the standard deviation of the diversity of the sites, whereas "Jackknife" calculates the jackknifed diversity for the entire data frame. Method "s" is not allowed for diversitycomp.
sortit	Sort the sites by increasing values of the diversity statistic.
digits	Number of digits in the results.
factor1	Variable of the environmental data frame that defines subsets to calculate diversity statistics for.
factor2	Optional second variable of the environmental data frame that defines subsets to calculate diversity statistics for in a crosstabulation with the other variable of the environmental data frame.
...	Other arguments passed to function diversityresult.

Details

These functions provide some alternative methods of obtaining results with diversity statistics, although functions `diversity`, `fisher.alpha`, `specpool`, `estimateR` and `specnumber` (all from **vegan**) are called to calculate the various statistics.

The reciprocal Berger-Parker diversity index is the reciprocal of the proportional abundance of the most dominant species.

J-evenness is calculated as: $H / \ln(S)$ where H is the Shannon diversity index and S the species richness.

E-evenness is calculated as: $\exp(H) / S$ where H is the Shannon diversity index and S the species richness.

The method of calculating the diversity statistics include following options: "all" calculates the diversity of the entire community (all sites pooled together), "s" calculates the diversity of each site separately, "mean" calculates the average diversity of the sites, whereas "Jackknife" calculates the jackknifed diversity for the entire data frame. Methods "s" and "mean" are not available for function diversitycomp. Gamma diversity estimators assume that the method is "all".

Functions diversityresult and diversitycomp allow to calculate diversity statistics for subsets of the community and environmental data sets. Function diversityresult calculates the diversity statistics for the specified level of a selected environmental variable. Function diversitycomp calculates the diversity statistics for all levels of a selected environmental variable separately. When a second environmental variable is provided, function diversitycomp calculates diversity statistics as a crosstabulation of both variables.

Value

The functions provide alternative methods of obtaining diversity results. For function diversitycomp, the number of sites is provided as "n".

Author(s)

Roeland Kindt (World Agroforestry Centre)

References

Kindt, R. & Coe, R. (2005) Tree diversity analysis: A manual and software for common statistical methods for ecological and biodiversity studies.

<http://www.worldagroforestry.org/resources/databases/tree-diversity-analysis>

Examples

```
library(vegan)
data(dune.env)
data(dune)
Diversity.1 <- diversityresult(dune, y=dune.env, factor='Management',
  level='NM', index='Shannon', method='s', sortit=TRUE, digits=3)
Diversity.1
diversitycomp(dune, y=dune.env, factor1='Management', factor2="Moisture",
  index='Shannon', method='all', sortit=TRUE, digits=3)
```

ensemble.batch

*Suitability mapping based on ensembles of modelling algorithms:
batch processing*

Description

The main function allows for batch processing of different species and different environmental RasterStacks. The function makes internal calls to [ensemble.test.splits](#), [ensemble.test](#) and [ensemble.raster](#).

Usage

```

ensemble.batch(x = NULL, xn = c(x), ext = NULL,
  species.presence = NULL, species.absence = NULL,
  presence.min = 20,
  an = 1000, excludep = FALSE,
  k.splits = 5, k.test = 0,
  n.ensembles = 1,
  SINK = FALSE,
  RASTER.format = "raster", RASTER.datatype = "INT2S", RASTER.NAflag = -32767,
  KML.out = FALSE, KML.maxpixels = 100000, KML.blur = 10,
  models.save = FALSE,
  threshold.method = "spec_sens", threshold.sensitivity = 0.9,
  ENSEMBLE.best = 0, ENSEMBLE.min = 0.7, ENSEMBLE.exponent = 1,
  input.weights = NULL,
  MAXENT = 1, GBM = 1, GBMSTEP = 1, RF = 1, GLM = 1, GLMSTEP = 1, GAM = 1,
  GAMSTEP = 1, MGCV = 1, MGCVFIX = 0, EARTH = 1, RPART = 1, NNET = 1,
  FDA = 1, SVM = 1, SVME = 1, BIOCLIM = 1, DOMAIN = 1, MAHAL = 1,
  PROBIT = FALSE, AUC.weights = TRUE,
  Yweights = "BIOMOD",
  layer.drops = NULL, factors = NULL, dummy.vars = NULL,
  formulae.defaults = TRUE, maxit = 100,
  MAXENT.a = NULL, MAXENT.an = 10000, MAXENT.BackData = NULL,
  MAXENT.path = paste(getwd(), "/models/maxent", sep=""),
  GBM.formula = NULL, GBM.n.trees = 2001,
  GBMSTEP.gbm.x = 2:(1 + raster::nlayers(x)),
  GBMSTEP.tree.complexity = 5, GBMSTEP.learning.rate = 0.005,
  GBMSTEP.bag.fraction = 0.5, GBMSTEP.step.size = 100,
  RF.formula = NULL, RF.ntree = 751, RF.mtry = floor(sqrt(raster::nlayers(x))),
  GLM.formula = NULL, GLM.family = binomial(link = "logit"),
  GLMSTEP.steps = 1000, STEP.formula = NULL, GLMSTEP.scope = NULL, GLMSTEP.k = 2,
  GAM.formula = NULL, GAM.family = binomial(link = "logit"),
  GAMSTEP.steps = 1000, GAMSTEP.scope = NULL, GAMSTEP.pos = 1,
  MGCV.formula = NULL, MGCV.select = FALSE,
  MGCVFIX.formula = NULL,
  EARTH.formula = NULL,
  EARTH.glm = list(family = binomial(link = "logit"), maxit = maxit),
  RPART.formula = NULL, RPART.xval = 50,
  NNET.formula = NULL, NNET.size = 8, NNET.decay = 0.01,
  FDA.formula = NULL,
  SVM.formula = NULL, SVME.formula = NULL,
  MAHAL.shape = 1)

ensemble.mean(RASTER.species.name = "Species001", RASTER.stack.name = "base",
  positive.filters = c("grd", "_ENSEMBLE_"), negative.filters = c("xml"),
  RASTER.format = "raster", RASTER.datatype = "INT2S", RASTER.NAflag = -32767,
  KML.out = FALSE, KML.maxpixels = 100000, KML.blur = 10,
  p = NULL, a = NULL,
  pt = NULL, at = NULL,

```

```
threshold = -1,
threshold.method = "spec_sens", threshold.sensitivity = 0.9)
```

Arguments

x	RasterStack object (stack) containing all layers to calibrate an ensemble.
xn	RasterStack object (stack) containing all layers that correspond to explanatory variables of an ensemble calibrated earlier with x. Several RasterStack objects can be provided in a format as c(stack1, stack2, stack3); these will be used sequentially. See also predict .
ext	an Extent object to limit the prediction to a sub-region of xn and the selection of background points to a sub-region of x, typically provided as c(lonmin, lonmax, latmin, latmax); see also predict , randomPoints and extent
species.presence	presence points used for calibrating the suitability models, available in 3-column (species, x, y) or (species, lon, lat) dataframe
species.absence	background points used for calibrating the suitability models, either available in a 3-column (species, x, y) or (species, lon, lat), or available in a 2-column (x, y) or (lon, lat) dataframe. In case of a 2-column dataframe, the same background locations will be used for all species.
presence.min	minimum number of presence locations for the organism (if smaller, no models are fitted).
an	number of background points for calibration to be selected with randomPoints in case argument a or species.absence is missing
excludep	parameter that indicates (if TRUE) that presence points will be excluded from the background points; see also randomPoints
k	If larger than 1, the number of groups to split between calibration (k-1) and evaluation (1) data sets (for example, k=5 results in 4/5 of presence and background points to be used for calibrating the models, and 1/5 of presence and background points to be used for evaluating the models). See also kfold .
k.splits	If larger than 1, the number of splits for the ensemble.test.splits step in batch processing. See also kfold .
k.test	If larger than 1, the number of groups to split between calibration (k-1) and evaluation (1) data sets when calibrating the final models (for example, k=5 results in 4/5 of presence and background points to be used for calibrating the models, and 1/5 of presence and background points to be used for evaluating the models). See also kfold .
n.ensembles	If larger than 1, the number of different ensembles generated per species in batch processing.
SINK	Append the results to a text file in subfolder 'outputs' (if TRUE). The name of file is based on species names. In case a file already exists, then results are appended. See also sink .
RASTER.format	Format of the raster files that will be generated. See writeFormats and writeRaster .

RASTER.datatype	Format of the raster files that will be generated. See dataType and writeRaster .
RASTER.NAflag	Value that is used to store missing data. See writeRaster .
KML.out	if FALSE, then no kml layers (layers that can be shown in Google Earth) are produced. If TRUE, then kml files will be saved in a subfolder 'kml'.
KML.maxpixels	Maximum number of pixels for the PNG image that will be displayed in Google Earth. See also KML .
KML.blur	Integer that results in increasing the size of the PNG image by $KML.blur^2$, which may help avoid blurring of isolated pixels. See also KML .
models.save	Save the list with model details to a file (if TRUE). The filename will be <code>species.name</code> with extension <code>.models</code> ; this file will be saved in subfolder of <code>models</code> . When loading this file, model results will be available as <code>ensemble.models</code> .
threshold.method	Method to calculate the threshold between predicted absence and presence; possibilities include <code>spec_sens</code> (highest sum of the true positive rate and the true negative rate), <code>kappa</code> (highest kappa value), <code>no_omission</code> (highest threshold that corresponds to no omission), <code>prevalence</code> (modeled prevalence is closest to observed prevalence) and <code>equal_sens_spec</code> (equal true positive rate and true negative rate). See threshold . Options specific to the BiodiversityR implementation are: <code>threshold.mean</code> (resulting in calculating the mean value of <code>spec_sens</code> , <code>equal_sens_spec</code> and <code>prevalence</code>) and <code>threshold.min</code> (resulting in calculating the minimum value of <code>spec_sens</code> , <code>equal_sens_spec</code> and <code>prevalence</code>).
threshold.sensitivity	Sensitivity value for <code>threshold.method = 'sensitivity'</code> . See threshold .
ENSEMBLE.best	The number of individual suitability models to be used in the consensus suitability map (based on a weighted average). In case this parameter is smaller than 1 or larger than the number of positive input weights of individual models, then all individual suitability models with positive input weights are included in the consensus suitability map.
ENSEMBLE.min	The minimum input weight (assumed to correspond to evaluation indices of suitability models) for a model to be included in the ensemble.
ENSEMBLE.exponent	Exponent applied to AUC values to convert AUC values into weights (for example, an exponent of 2 converts input weights of 0.7, 0.8 and 0.9 into $0.7^2=0.49$, $0.8^2=0.64$ and $0.9^2=0.81$). See ensemble.test for details.
input.weights	array with numeric values for the different modelling algorithms; if NULL then values provided by parameters such as <code>MAXENT</code> and <code>GBM</code> will be used. As an alternative, the output from <code>ensemble.test.splits</code> can be used.
MAXENT	Input weight for a maximum entropy model (maxent). (Only weights > 0 will be used.)
GBM	Input weight for a boosted regression trees model (gbm). (Only weights > 0 will be used.)
GBMSTEP	Input weight for a stepwise boosted regression trees model (gbm.step). (Only weights > 0 will be used.)

RF	Input weight for a random forest model (randomForest). (Only weights > 0 will be used.)
GLM	Input weight for a generalized linear model (glm). (Only weights > 0 will be used.)
GLMSTEP	Input weight for a stepwise generalized linear model (stepAIC). (Only weights > 0 will be used.)
GAM	Input weight for a generalized additive model (gam). (Only weights > 0 will be used.)
GAMSTEP	Input weight for a stepwise generalized additive model (step.gam). (Only weights > 0 will be used.)
MGCV	Input weight for a generalized additive model (gam). (Only weights > 0 will be used.)
MGCVFIX	number: if larger than 0, then a generalized additive model with fixed d.f. regression splines (gam) will be fitted among ensemble
EARTH	Input weight for a multivariate adaptive regression spline model (earth). (Only weights > 0 will be used.)
RPART	Input weight for a recursive partitioning and regression tree model (rpart). (Only weights > 0 will be used.)
NNET	Input weight for an artificial neural network model (nnet). (Only weights > 0 will be used.)
FDA	Input weight for a flexible discriminant analysis model (fda). (Only weights > 0 will be used.)
SVM	Input weight for a support vector machine model (ksvm). (Only weights > 0 will be used.)
SVME	Input weight for a support vector machine model (svm). (Only weights > 0 will be used.)
BIOCLIM	Input weight for the BIOCLIM algorithm (bioclim). (Only weights > 0 will be used.)
DOMAIN	Input weight for the DOMAIN algorithm (domain). (Only weights > 0 will be used.)
MAHAL	Input weight for the Mahalonobis algorithm (mahal). (Only weights > 0 will be used.)
PROBIT	If TRUE, then subsequently to the fitting of the individual algorithm (e.g. maximum entropy or GAM) a generalized linear model (glm) with <code>probit</code> link family= <code>binomial(link="probit")</code> will be fitted to transform the predictions, using the previous predictions as explanatory variable. This transformation results in all model predictions to be probability estimates.
AUC.weights	If TRUE, then use the average of the AUC for the different submodels in the different crossvalidation runs as weights for the 'full' ensemble model. See ensemble.test.splits for details.
Yweights	chooses how cases of presence and background (absence) are weighted; "BIOMOD" results in equal weighting of all presence and all background cases, "equal" results in equal weighting of all cases. The user can supply a vector of weights similar to the number of cases in the calibration data set.

<code>layer.drops</code>	vector that indicates which layers should be removed from RasterStack <code>x</code> . See also addLayer .
<code>factors</code>	vector that indicates which variables are factors; see also prepareData
<code>dummy.vars</code>	vector that indicates which variables are dummy variables (influences formulae suggestions)
<code>formulae.defaults</code>	Suggest formulae for most of the models (if TRUE). See also ensemble.formulae .
<code>maxit</code>	Maximum number of iterations for some of the models. See also glm.control , gam.control and nnet .
<code>MAXENT.a</code>	background points used for calibrating the maximum entropy model (maxent), typically available in 2-column (lon, lat) dataframe; see also prepareData and extract . Ignored if <code>MAXENT.BackData</code> is provided.
<code>MAXENT.an</code>	number of background points for calibration to be selected with randomPoints in case argument <code>MAXENT.a</code> is missing. When used with the <code>ensemble.batch</code> function, the same background locations will be used for each of the species runs; this implies that for each species, presence locations are not excluded from the background data for this function.
<code>MAXENT.BackData</code>	dataframe containing explanatory variables for the background locations. This information will be used for calibrating the maximum entropy model (maxent). When used with the <code>ensemble.batch</code> function, the same background locations will be used for each of the cross-validation runs; this is based on the assumption that a large number (~10000) of background locations are used.
<code>MAXENT.path</code>	path to the directory where output files of the maximum entropy model are stored; see also maxent
<code>GBM.formula</code>	formula for the boosted regression trees algorithm; see also gbm
<code>GBM.n.trees</code>	total number of trees to fit for the boosted regression trees model; see also gbm
<code>GBMSTEP.gbm.x</code>	indices of column numbers with explanatory variables for stepwise boosted regression trees; see also gbm.step
<code>GBMSTEP.tree.complexity</code>	complexity of individual trees for stepwise boosted regression trees; see also gbm.step
<code>GBMSTEP.learning.rate</code>	weight applied to individual trees for stepwise boosted regression trees; see also gbm.step
<code>GBMSTEP.bag.fraction</code>	proportion of observations used in selecting variables for stepwise boosted regression trees; see also gbm.step
<code>GBMSTEP.step.size</code>	number of trees to add at each cycle for stepwise boosted regression trees (should be small enough to result in a smaller holdout deviance than the initial number of trees [50]); see also gbm.step
<code>RF.formula</code>	formula for the random forest algorithm; see also randomForest
<code>RF.ntree</code>	number of trees to grow for random forest algorithm; see also randomForest

RF.mtry	number of variables randomly sampled as candidates at each split for random forest algorithm; see also randomForest
GLM.formula	formula for the generalized linear model; see also glm
GLM.family	description of the error distribution and link function for the generalized linear model; see also glm
GLMSTEP.steps	maximum number of steps to be considered for stepwise generalized linear model; see also stepAIC
STEP.formula	formula for the "starting model" to be considered for stepwise generalized linear model; see also stepAIC
GLMSTEP.scope	range of models examined in the stepwise search; see also stepAIC
GLMSTEP.k	multiple of the number of degrees of freedom used for the penalty (only $k = 2$ gives the genuine AIC); see also stepAIC
GAM.formula	formula for the generalized additive model; see also gam
GAM.family	description of the error distribution and link function for the generalized additive model; see also gam
GAMSTEP.steps	maximum number of steps to be considered in the stepwise generalized additive model; see also step.gam
GAMSTEP.scope	range of models examined in the step-wise search n the stepwise generalized additive model; see also step.gam
GAMSTEP.pos	parameter expected to be set to 1 to allow for fitting of the stepwise generalized additive model
MGCV.formula	formula for the generalized additive model; see also gam
MGCV.select	if TRUE, then the smoothing parameter estimation that is part of fitting can completely remove terms from the model; see also gam
MGCVFIX.formula	formula for the generalized additive model with fixed d.f. regression splines; see also gam (the default formulae sets "s(..., fx=TRUE, ...)"; see also s)
EARTH.formula	formula for the multivariate adaptive regression spline model; see also earth
EARTH.glm	list of arguments to pass on to glm ; see also earth
RPART.formula	formula for the recursive partitioning and regression tree model; see also rpart
RPART.xval	number of cross-validations for the recursive partitioning and regression tree model; see also rpart.control
NNET.formula	formula for the artificial neural network model; see also nnet
NNET.size	number of units in the hidden layer for the artificial neural network model; see also nnet
NNET.decay	parameter of weight decay for the artificial neural network model; see also nnet
FDA.formula	formula for the flexible discriminant analysis model; see also fda
SVM.formula	formula for the support vector machine model; see also ksvm
SVME.formula	formula for the support vector machine model; see also svm
MAHAL.shape	parameter that influences the transformation of output values of mahal . See details section.

<code>RASTER.species.name</code>	First part of the names of the raster files, expected to identify the modelled species (or organism).
<code>RASTER.stack.name</code>	Last part of the names of the raster files, expected to identify the predictor stack used.
<code>positive.filters</code>	vector that indicates parts of filenames for files that will be included in the calculation of the mean probability values
<code>negative.filters</code>	vector that indicates parts of filenames for files that will not be included in the calculation of the mean probability values
<code>p</code>	presence points used for calibrating the suitability models, typically available in 2-column (x, y) or (lon, lat) dataframe; see also prepareData and extract
<code>a</code>	background points used for calibrating the suitability models, typically available in 2-column (x, y) or (lon, lat) dataframe; see also prepareData and extract
<code>pt</code>	presence points used for evaluating the suitability models, typically available in 2-column (lon, lat) dataframe; see also prepareData
<code>at</code>	background points used for calibrating the suitability models, typically available in 2-column (lon, lat) dataframe; see also prepareData and extract
<code>threshold</code>	Threshold value that will be used to distinguish between presence and absence. If < 0 , then a threshold value will be calculated from the provided presence p and absence a locations.

Details

This function allows for batch processing of different species and different environmental Raster-Stacks. The function makes internal calls to [ensemble.test.splits](#), [ensemble.test](#) and [ensemble.raster](#).

[ensemble.test.splits](#) results in a cross-validation procedure whereby the data set is split in calibration and testing subsets and the best weights for the ensemble model are determined (including the possibility for weights = 0).

[ensemble.test](#) is the step whereby models are calibrated using all the available presence data.

[ensemble.raster](#) is the final step whereby raster layers are produced for the ensemble model.

Function `ensemble.mean` results in raster layers that are based on the summary of several ensemble layers: the new ensemble has probability values that are the mean of the probabilities of the different raster layers, the presence-absence threshold is derived for this new ensemble layer, whereas the count reflects the number of ensemble layers where presence was predicted. Note the assumption that input probabilities are scaled between 0 and 1000 (as the output from [ensemble.raster](#)), whereas thresholds are based on actual probabilities (scaled between 0 and 1).

Value

The function finally results in ensemble raster layers for each species, including the fitted values for the ensemble model, the estimated presence-absence and the count of the number of submodels prediction presence and absence.

Author(s)

Roeland Kindt (World Agroforestry Centre), Eike Luedeling (World Agroforestry Centre) and Evert Thomas (Bioversity International)

References

Buisson L, Thuiller W, Casajus N, Lek S and Grenouillet G. 2010. Uncertainty in ensemble forecasting of species distribution. *Global Change Biology* 16: 1145-1157

See Also

[ensemble.test.splits](#), [ensemble.test](#), [ensemble.raster](#)

Examples

```
## Not run:
# based on examples in the dismo package

# get predictor variables
library(dismo)
predictor.files <- list.files(path=paste(system.file(package="dismo"), '/ex', sep=''),
  pattern='grd', full.names=TRUE)
predictors <- stack(predictor.files)
# subset based on Variance Inflation Factors
predictors <- subset(predictors, subset=c("bio5", "bio6",
  "bio16", "bio17", "biome"))
predictors
predictors@title <- "base"

# presence points
presence_file <- paste(system.file(package="dismo"), '/ex/bradypus.csv', sep='')
pres <- read.table(presence_file, header=TRUE, sep=',')
pres[,1] <- rep("Bradypus", nrow(pres))

# choose background points
ext <- extent(-90, -32, -33, 23)
background <- randomPoints(predictors, n=1000, ext=ext, extf = 1.00)

# fit 3 ensembles with batch processing, choosing the best ensemble model based on the
# average AUC of 4-fold split of calibration and testing data
# final models use all available presence data and average weights determined by the
# ensemble.test.splits function (called internally)
# batch processing can handle several species by using 3-column species.presence and
# species.absence data sets
# note that these calculations can take a while

ensemble.nofactors <- ensemble.batch(x=predictors, ext=ext,
  species.presence=pres,
  species.absence=background,
  k.splits=4, k.test=0,
  n.ensembles=3,
  SINK=TRUE,
```

```

layer.drops=c("biome"),
ENSEMBLE.best=0, ENSEMBLE.exponent=c(1, 2, 4, 6, 8),
ENSEMBLE.min=0.7,
MAXENT=1, GBM=1, GBMSTEP=0, RF=1, GLM=1, GLMSTEP=1, GAM=1, GAMSTEP=0, MGCV=1,
EARTH=1, RPART=1, NNET=1, FDA=1, SVM=1, SVME=1, BIOCLIM=1, DOMAIN=1, MAHAL=0,
Yweights="BIOMOD",
formulae.defaults=TRUE)

# summaries for the 3 ensembles for the species
# summaries are based on files in folders ensemble, ensemble/presence and
# ensemble/count
pres1 <- pres[,-1]
ensemble.mean(RASTER.species.name="Bradypus",
              p=pres1, a=background,
              KML.out=T)

## End(Not run)

```

```
ensemble.dummy.variables
```

*Suitability mapping based on ensembles of modelling algorithms:
handling of categorical data*

Description

The basic function `ensemble.dummy.variables` creates new raster layers representing dummy variables (coded 0 or 1) for all or the most frequent levels of a categorical variable. Sometimes the creation of dummy variables is needed for proper handling of categorical data for some of the suitability modelling algorithms.

Usage

```

ensemble.dummy.variables(xcat=NULL,
                        freq.min=50, most.frequent=5,
                        overwrite=TRUE, ...)

ensemble.accepted.categories(xcat = NULL, categories = NULL,
                             filename=NULL, overwrite=TRUE, ...)

ensemble.simplified.categories(xcat = NULL, p = NULL,
                               filename=NULL, overwrite=TRUE, ...)

```

Arguments

`xcat` RasterLayer object ([raster](#)) containing values for a categorical explanatory variable.

freq.min	Minimum frequency for a dummy raster layer to be created for the corresponding factor level. See also freq .
most.frequent	Number of dummy raster layers to be created (if larger than 0), corresponding to the same number of most frequent factor levels See also freq .
overwrite	overwrite an existing file name with the same name (if TRUE). See also writeRaster .
...	additional arguments for writeRaster or (for ensemble.dummy.variables, writeRaster).
categories	numeric vector providing the accepted levels of a categorical raster layer; expected to correspond to the levels encountered during calibration
filename	name for the output file. See also writeRaster .
p	presence points that will be used for calibrating the suitability models, typically available in 2-column (x, y) or (lon, lat) dataframe; see also prepareData and extract

Details

The basic function `ensemble.dummy.variables` creates dummy variables from a `RasterLayer` object (see [raster](#)) that represents a categorical variable. With `freq.min` and `most.frequent` it is possible to limit the number of dummy variables that will be created. For example, `most.frequent = 5` results in five dummy variables to be created.

Function `ensemble.accepted.categories` modifies the `RasterLayer` object (see [raster](#)) by replacing cell values for categories (levels) that are not accepted with missing values.

Function `ensemble.simplified.categories` modifies the `RasterLayer` object (see [raster](#)) by replacing cell values for categories (levels) where none of the presence points occur with the same level. This new level is coded by the maximum coding level for these 'outside categories'.

Value

The basic function `ensemble.raster` mainly results in the creation of raster layers that correspond to dummy variables.

Author(s)

Roeland Kindt (World Agroforestry Centre) and Evert Thomas (Bioversity International)

See Also

[ensemble.test](#), [ensemble.raster](#)

Examples

```
## Not run:

# get predictor variables
library(dismo)
predictor.files <- list.files(path=paste(system.file(package="dismo"), '/ex', sep=''),
  pattern='grd', full.names=TRUE)
predictors <- stack(predictor.files)
```

```

biome.layer <- predictors[["biome"]]
biome.layer

# create dummy layers for the 5 most frequent factor levels

ensemble.dummy.variables(xcat=biome.layer, most.frequent=5,
  overwrite=TRUE)

# check whether dummy variables were created
predictor.files <- list.files(path=paste(system.file(package="dismo"), '/ex', sep=''),
  pattern='grd', full.names=TRUE)
predictors <- stack(predictor.files)
predictors
names(predictors)

# once dummy variables were created, avoid using the original categorical data layer
predictors <- subset(predictors, subset=c("bio5", "bio6", "bio16", "bio17",
  "biome_1", "biome_2", "biome_7", "biome_8", "biome_13"))
predictors
predictors@title <- "base"

# presence points
presence_file <- paste(system.file(package="dismo"), '/ex/bradypus.csv', sep='')
pres <- read.table(presence_file, header=TRUE, sep=',')[,-1]

# the kfold function randomly assigns data to groups;
# groups are used as calibration (1/5) and training (4/5) data
grouppp <- kfold(pres, 5)
pres_train <- pres[grouppp != 1, ]
pres_test <- pres[grouppp == 1, ]

# choose background points
ext <- extent(-90, -32, -33, 23)
background <- randomPoints(predictors, n=1000, ext=ext, extf=1.00)
colnames(background)=c('lon', 'lat')
groupa <- kfold(background, 5)
backg_train <- background[groupa != 1, ]
backg_test <- background[groupa == 1, ]

# fit four ensemble models (RF, GLM, BIOCLIM, DOMAIN)
# note that dummy variables are not used for BIOCLIM and DOMAIN
# (neither are categorical variables)
ensemble.nofactors <- ensemble.test(x=predictors, p=pres_train, a=backg_train,
  pt=pres_test, at=backg_test,
  species.name="Bradypus",
  VIF=T,
  MAXENT=1, GBM=1, GBMSTEP=1, RF=1, GLM=1, GLMSTEP=1, GAM=1,
  GAMSTEP=1, MGCV=1, MGCVFIX=1, EARTH=1, RPART=1, NNET=1, FDA=1,
  SVM=1, SVME=1, BIOCLIM=1, DOMAIN=1, MAHAL=0,
  Yweights="BIOMOD",
  dummy.vars=c("biome_1", "biome_2", "biome_7", "biome_8", "biome_13"),
  PLOTS=FALSE, evaluations.keep=TRUE)

```

```
## End(Not run)
```

ensemble.raster	<i>Suitability mapping based on ensembles of modelling algorithms: consensus mapping</i>
-----------------	--

Description

The basic function `ensemble.raster` creates two consensus raster layers, one based on a (weighted) average of different suitability modelling algorithms, and a second one documenting the number of modelling algorithms that predict presence of the focal organisms. Modelling algorithms include maximum entropy (MAXENT), boosted regression trees, random forests, generalized linear models (including stepwise selection of explanatory variables), generalized additive models (including stepwise selection of explanatory variables), multivariate adaptive regression splines, regression trees, artificial neural networks, flexible discriminant analysis, support vector machines, the BIOCLIM algorithm, the DOMAIN algorithm and the Mahalanobis algorithm. These sets of functions were developed in parallel with the `biomod2` package, especially for inclusion of the maximum entropy algorithm, but also to allow for a more direct integration with the `BiodiversityR` package, more direct handling of model formulae and greater focus on mapping. Researchers and students of species distribution are strongly encouraged to familiarize themselves with all the options of the `biomod2` and `dismo` packages.

Usage

```
ensemble.raster(xn = NULL, ext = NULL,
               models.list = NULL,
               input.weights = models.list$output.weights,
               thresholds = models.list$thresholds,
               RASTER.species.name = "Species001", RASTER.stack.name = xn@title,
               RASTER.format = "raster", RASTER.datatype = "INT2S", RASTER.NAflag = -32767,
               RASTER.models.override = TRUE,
               KML.out = FALSE, KML.maxpixels = 100000, KML.blur = 10,
               evaluate = FALSE, SINK = FALSE,
               p = models.list$p, a = models.list$a,
               pt = models.list$pt, at = models.list$at)

ensemble.habitat.change(base.map=file.choose(),
                       other.maps=choose.files(),
                       change.folder="ensembles/change",
                       RASTER.format = "raster", RASTER.datatype = "INT1U", RASTER.NAflag = 255,
                       KML.out = FALSE, KML.folder = "kml/change",
                       KML.maxpixels = 100000, KML.blur = 10)
```

Arguments

<code>xn</code>	RasterStack object (stack) containing all layers that correspond to explanatory variables of an ensemble calibrated earlier with ensemble.test . See also predict .
<code>ext</code>	an Extent object to limit the prediction to a sub-region of <code>xn</code> and the selection of background points to a sub-region of <code>x</code> , typically provided as <code>c(lonmin, lonmax, latmin, latmax)</code> ; see also predict , randomPoints and extent
<code>models.list</code>	list with 'old' model objects such as MAXENT or RF.
<code>input.weights</code>	array with numeric values for the different modelling algorithms; if NULL then values provided by parameters such as MAXENT and GBM will be used. As an alternative, the output from <code>ensemble.test.splits</code> can be used.
<code>thresholds</code>	array with the threshold values separating predicted presence for each of the algorithms.
<code>RASTER.species.name</code>	First part of the names of the raster files that will be generated, expected to identify the modelled species (or organism).
<code>RASTER.stack.name</code>	Last part of the names of the raster files that will be generated, expected to identify the predictor stack used.
<code>RASTER.format</code>	Format of the raster files that will be generated. See writeFormats and writeRaster .
<code>RASTER.datatype</code>	Format of the raster files that will be generated. See dataType and writeRaster .
<code>RASTER.NAflag</code>	Value that is used to store missing data. See writeRaster .
<code>RASTER.models.overwrite</code>	Overwrite the raster files that correspond to each suitability modelling algorithm (if TRUE). (Overwriting actually implies that raster files are created or overwritten that start with "working_").
<code>KML.out</code>	if FALSE, then no kml layers (layers that can be shown in Google Earth) are produced. If TRUE, then kml files will be saved in a subfolder 'kml'.
<code>KML.maxpixels</code>	Maximum number of pixels for the PNG image that will be displayed in Google Earth. See also KML .
<code>KML.blur</code>	Integer that results in increasing the size of the PNG image by $KML.blur^2$, which may help avoid blurring of isolated pixels. See also KML .
<code>evaluate</code>	if TRUE, then evaluate the created raster layers at locations <code>p</code> , <code>a</code> , <code>pt</code> and <code>at</code> (if provided). See also evaluate
<code>SINK</code>	Append the results to a text file in subfolder 'outputs' (if TRUE). The name of file is based on argument <code>RASTER.species.name</code> . In case the file already exists, then results are appended. See also sink .
<code>p</code>	presence points used for calibrating the suitability models, typically available in 2-column (x, y) or (lon, lat) dataframe; see also prepareData and extract
<code>a</code>	background points used for calibrating the suitability models, typically available in 2-column (x, y) or (lon, lat) dataframe; see also prepareData and extract

pt	presence points used for evaluating the suitability models, typically available in 2-column (lon, lat) dataframe; see also prepareData
at	background points used for calibrating the suitability models, typically available in 2-column (lon, lat) dataframe; see also prepareData and extract
base.map	filename with baseline map used to produce maps that show changes in suitable habitat
other.maps	files with other maps used to produce maps that show changes in suitable habitat from a defined base.map
change.folder	folder where new maps documenting changes in suitable habitat will be stored. NOTE: please ensure that the base folder (eg: ../ensembles) exists already.
KML.folder	folder where new maps (in Google Earth format) documenting changes in suitable habitat will be stored. NOTE: please ensure that the base folder (eg: ../kml) exists already.

Details

The basic function `ensemble.raster` fits individual suitability models for all models with positive input weights. In subfolder "models" of the working directory, suitability maps for the individual suitability modelling algorithms are stored. In subfolder "ensembles", a consensus suitability map based on a weighted average of individual suitability models is stored. In subfolder "ensembles/presence", a presence-absence (1-0) map will be provided. In subfolder "ensembles/count", a consensus suitability map based on the number of individual suitability models that predict presence of the focal organism is stored.

Several of the features of `ensemble.raster` are also available from `ensemble.test`. The main difference between the two functions is that `ensemble.raster` generates raster layers for individual suitability models, whereas the purpose of `ensemble.test` is specifically to test different suitability modelling algorithms.

Note that values in suitability maps are integer values that were calculated by multiplying probabilities by 1000 (see also [trunc](#)).

As the Mahalanobis function ([mahal](#)) does not always provide values within a range of 0 - 1, the output values are rescaled by first subtracting the value of $1 - \text{MAHAL}.\text{shape}$ from each prediction, followed by calculating the absolute value, followed by calculating the reciprocal value and finally multiplying this reciprocal value with $\text{MAHAL}.\text{shape}$. As this rescaling method does not estimate probabilities, inclusion in the calculation of a (weighted) average of ensemble probabilities may be problematic (the same applies to other distance-based methods).

The `ensemble.habitat.change` function produces new raster layers that show changes in suitable and not suitable habitat between a base raster and a list of other rasters. The output uses the following coding: 0 = areas that remain unsuitable, 11 = areas that remain suitable, 10 = areas of lost habitat, 1 = areas of new habitat. (Codes are inspired on a binary classification of habitat suitability in base [1- or 0-] and other layer [-1 or -0], eg new habitat is coded 01 = 1).

With `KML.out = TRUE`, kml files are created in a subfolder named "KML". The colouring of the consensus suitability PNG is based on 20 intervals of size 50 between 0 and 1000. The colouring of the presence-absence PNG uses green for presence and red for absence. The colouring of the count suitability PNG uses black for zero (no models predict presence) and blue for the theoretical maximum number of models to predict presence (i.e. the count of all final weights), whereas intermediate numbers (1 to theoretical maximum - 1) are ranged from red to green. The colouring of the

habitat change maps are: black (cells that are never suitable [value: 0]), green (cells that are always suitable [value: 11]), red (cells that are lost habitat [value: 10] and blue (cells that are new habitat [value: 1]).

Value

The basic function `ensemble.raster` mainly results in the creation of raster layers that correspond to fitted probabilities of presence of individual suitability models (in folder "models") and consensus models (in folder "ensembles"), and the number of suitability models that predict presence (in folder "ensembles"). Prediction of presence is based on a threshold usually defined by maximizing the sum of the true presence and true absence rates (see `threshold.method` and also [ModelEvaluation](#)).

If desired by the user, the `ensemble.raster` function also saves details of fitted suitability models or data that can be plotted with the [evaluation.strip.plot](#) function.

Author(s)

Roeland Kindt (World Agroforestry Centre), Eike Luedeling (World Agroforestry Centre) and Evert Thomas (Bioversity International)

References

Buisson L, Thuiller W, Casajus N, Lek S and Grenouillet G. 2010. Uncertainty in ensemble forecasting of species distribution. *Global Change Biology* 16: 1145-1157

See Also

[evaluation.strip.plot](#), [ensemble.test](#), [ensemble.test.splits](#), [ensemble.batch](#)

Examples

```
## Not run:
# based on examples in the dismo package

# get predictor variables
library(dismo)
predictor.files <- list.files(path=paste(system.file(package="dismo"), '/ex', sep=''),
  pattern='grd', full.names=TRUE)
predictors <- stack(predictor.files)
# subset based on Variance Inflation Factors
predictors <- subset(predictors, subset=c("bio5", "bio6",
  "bio16", "bio17", "biome"))
predictors
predictors
predictors@title <- "base"

# presence points
# presence points
presence_file <- paste(system.file(package="dismo"), '/ex/bradypus.csv', sep='')
pres <- read.table(presence_file, header=TRUE, sep=',')[,-1]
```

```

# choose background points
ext <- extent(-90, -32, -33, 23)
background <- randomPoints(predictors, n=1000, ext=ext, extf = 1.00)

# if desired, change working directory where subfolders of "models" and
# "ensembles" will be created
# raster layers will be saved in subfolders of /models and /ensembles:
getwd()

# first calibrate the ensemble
# calibration is done in two steps
# in step 1, a k-fold procedure is used to determine the weights
# in step 2, models are calibrated for all presence and background locations
# factor is not used as it is not certain whether correct levels will be used
# it may therefore be better to use dummy variables

# step 1: 4-fold cross-validation
ensemble.calibrate.step1 <- ensemble.test.splits(x=predictors, p=pres, a=background,
  ext=ext,
  k=4,
  layer.drops=c("biome"),
  SINK=TRUE, species.name="Bradypus",
  MAXENT=1, GBM=1, GBMSTEP=0, RF=1, GLM=1, GLMSTEP=1, GAM=1,
  GAMSTEP=1, MGCV=1, MGCVFIX=1, EARTH=1, RPART=1, NNET=1, FDA=1,
  SVM=1, SVME=1, BIOCLIM=1, DOMAIN=1, MAHAL=0,
  ENSEMBLE.tune=TRUE, PROBIT=TRUE,
  ENSEMBLE.best=0, ENSEMBLE.exponent=c(1, 2, 4, 6, 8),
  ENSEMBLE.min=0.7,
  Yweights="BIOMOD", factors=c("biome"),
  PLOTS=FALSE, formulae.defaults=TRUE,
  GBMSTEP.learning.rate=0.002)

# step 2: create the models that will be used for the raster predictions
# models with input.weights < 0.05 are excluded

output.weights <- ensemble.calibrate.step1$output.weights.AUC
output.weights[output.weights < 0.05] <- 0

ensemble.calibrate.step2 <- ensemble.test(x=predictors, p=pres, a=background,
  ext=ext,
  layer.drops=c("biome"),
  SINK=TRUE, species.name="Bradypus",
  models.keep=TRUE,
  input.weights=output.weights,
  AUC.weights=FALSE, ENSEMBLE.tune=FALSE, PROBIT=TRUE,
  Yweights="BIOMOD", factors=c("biome"),
  PLOTS=FALSE, formulae.defaults=TRUE,
  GBMSTEP.learning.rate=0.002)

# step 3: use previously calibrated models
ensemble.nofactors1 <- ensemble.raster(xn=predictors, ext=ext,
  models.list=ensemble.calibrate.step2$models,
  input.weights=output.weights,

```

```

    thresholds=ensemble.calibrate.step2$models$thresholds,
    SINK=TRUE,
    RASTER.species.name="Bradypus", RASTER.stack.name="base")

# use the base map to check for changes in suitable habitat
# this type of analysis is typically done with different predictor layers
# (for example, predictor layers representing different possible future climates)
# in this example, changes from a previous model (ensemble.nofactors1)
# are contrasted with a newly calibrated model (ensemble.nofactors2)
ensemble.calibrate.step1 <- ensemble.test.splits(x=predictors, p=pres, a=background,
  ext=ext,
  k=4,
  layer.drops=c("biome"),
  SINK=TRUE, species.name="Bradypus",
  MAXENT=1, GBM=1, GBMSTEP=0, RF=1, GLM=1, GLMSTEP=1, GAM=1,
  GAMSTEP=1, MGCV=1, MGCVFIX=1, EARTH=1, RPART=1, NNET=1, FDA=1,
  SVM=1, SVME=1, BIOCLIM=1, DOMAIN=1, MAHAL=0,
  ENSEMBLE.tune=TRUE, PROBIT=TRUE,
  ENSEMBLE.best=0, ENSEMBLE.exponent=c(1, 2, 4, 6, 8),
  ENSEMBLE.min=0.7,
  Yweights="BIOMOD",
  PLOTS=FALSE, formulae.defaults=TRUE,
  GBMSTEP.learning.rate=0.002)

output.weights <- ensemble.calibrate.step1$output.weights.AUC
output.weights[output.weights < 0.05] <- 0

ensemble.calibrate.step2 <- ensemble.test(x=predictors, p=pres, a=background,
  ext=ext,
  layer.drops=c("biome"),
  SINK=TRUE, species.name="Bradypus",
  models.keep=TRUE,
  input.weights=output.weights,
  AUC.weights=FALSE, ENSEMBLE.tune=FALSE, PROBIT=TRUE,
  Yweights="BIOMOD",
  PLOTS=FALSE, formulae.defaults=TRUE,
  GBMSTEP.learning.rate=0.002)

ensemble.nofactors2 <- ensemble.raster(xn=predictors, ext=ext,
  models.list=ensemble.calibrate.step2$models,
  input.weights=output.weights,
  thresholds=ensemble.calibrate.step2$models$thresholds,
  SINK=TRUE,
  RASTER.species.name="Bradypus", RASTER.stack.name="recalibrated")

base.file <- paste(getwd(), "/ensembles/presence/Bradypus_base.grd", sep="")
other.file <- paste(getwd(), "/ensembles/presence/Bradypus_recalibrated.grd", sep="")

changed.habitat <- ensemble.habitat.change(base.map=base.file,
  other.maps=c(other.file),
  change.folder="ensembles/change")

## End(Not run)

```

ensemble.test	<i>Suitability mapping based on ensembles of modelling algorithms: comparison of different algorithms and calibration</i>
---------------	---

Description

The basic function `ensemble.test` allows to evaluate different algorithms for (species) suitability modelling, including maximum entropy (MAXENT), boosted regression trees, random forests, generalized linear models (including stepwise selection of explanatory variables), generalized additive models (including stepwise selection of explanatory variables), multivariate adaptive regression splines, regression trees, artificial neural networks, flexible discriminant analysis, support vector machines, the BIOCLIM algorithm, the DOMAIN algorithm and the Mahalanobis algorithm. These sets of functions were developed in parallel with the `biomod2` package, especially for inclusion of the maximum entropy algorithm, but also to allow for a more direct integration with the `BiodiversityR` package, more direct handling of model formulae and greater focus on mapping. Researchers and students of species distribution are strongly encouraged to familiarize themselves with all the options of the `BIOMOD` and `dismo` packages.

Usage

```
ensemble.test(x = NULL, p = NULL, a = NULL, an = 1000, excludep = FALSE, ext = NULL,
             k = 0, pt = NULL, at = NULL,
             TrainData = NULL, TestData = NULL,
             VIF = FALSE, COR = FALSE,
             SINK = FALSE, PLOTS = TRUE,
             threshold.method = "spec_sens", threshold.sensitivity = 0.9,
             evaluations.keep = FALSE,
             models.list = NULL, models.keep = FALSE,
             models.save = FALSE, species.name = "Species001",
             AUC.weights = TRUE, ENSEMBLE.tune = FALSE,
             ENSEMBLE.best = 0, ENSEMBLE.min = 0.7, ENSEMBLE.exponent = 1,
             input.weights = NULL,
             MAXENT = 1, GBM = 1, GBMSTEP = 1, RF = 1, GLM = 1, GLMSTEP = 1,
             GAM = 1, GAMSTEP = 1, MGCV = 1, MGCVFIX = 0, EARTH = 1,
             RPART = 1, NNET = 1, FDA = 1, SVM = 1, SVME = 1,
             BIOCLIM = 1, DOMAIN = 1, MAHAL = 1,
             GEODIST = 0,
             PROBIT = FALSE,
             Yweights = "BIOMOD",
             layer.drops = NULL, factors = NULL, dummy.vars = NULL,
             formulae.defaults = TRUE, maxit = 100,
             MAXENT.a = NULL, MAXENT.an = 10000, MAXENT.BackData = NULL,
             MAXENT.path=paste(getwd(), "/models/maxent_", species.name, sep=""),
             GBM.formula = NULL, GBM.n.trees = 2001,
             GBMSTEP.gbm.x = 2:(ncol(TrainData.vars)+1), GBMSTEP.tree.complexity = 5,
```

```

GBMSTEP.learning.rate = 0.005, GBMSTEP.bag.fraction = 0.5,
GBMSTEP.step.size = 100,
RF.formula = NULL, RF.ntree = 751,
RF.mtry = floor(sqrt(ncol(TrainData.vars))),
GLM.formula = NULL, GLM.family = binomial(link = "logit"),
GLMSTEP.steps = 1000, STEP.formula = NULL, GLMSTEP.scope = NULL,
GLMSTEP.k = 2,
GAM.formula = NULL, GAM.family = binomial(link = "logit"),
GAMSTEP.steps = 1000, GAMSTEP.scope = NULL, GAMSTEP.pos = 1,
MGCV.formula = NULL, MGCV.select = FALSE,
MGCVFIX.formula = NULL,
EARTH.formula = NULL,
EARTH.glm = list(family = binomial(link = "logit"), maxit = maxit),
RPART.formula = NULL, RPART.xval = 50,
NNET.formula = NULL, NNET.size = 8, NNET.decay = 0.01,
FDA.formula = NULL,
SVM.formula = NULL,
SVME.formula = NULL,
MAHAL.shape = 1,
RASTER.format = "raster")

ensemble.test.splits(x = NULL, p = NULL, a = NULL, an = 1000,
  excludep = FALSE, ext = NULL,
  k = 5,
  TrainData = NULL,
  VIF = FALSE, COR = FALSE,
  SINK = FALSE, PLOTS = FALSE,
  data.keep = FALSE,
  species.name = "Species001",
  threshold.method = "spec_sens", threshold.sensitivity = 0.9,
  AUC.weights = TRUE, ENSEMBLE.tune = FALSE,
  ENSEMBLE.best = 0, ENSEMBLE.min = 0.7, ENSEMBLE.exponent = 1,
  input.weights = NULL,
  MAXENT = 1, GBM = 1, GBMSTEP = 1, RF = 1, GLM = 1, GLMSTEP = 1,
  GAM = 1, GAMSTEP = 1, MGCV = 1, MGCVFIX = 0, EARTH = 1,
  RPART = 1, NNET = 1, FDA = 1, SVM = 1, SVME = 1,
  BIOCLIM = 1, DOMAIN = 1, MAHAL = 1,
  PROBIT = FALSE,
  Yweights = "BIOMOD",
  layer.drops = NULL, factors = NULL, dummy.vars = NULL,
  formulae.defaults = TRUE, maxit = 100,
  MAXENT.a = NULL, MAXENT.an = 10000, MAXENT.BackData = NULL,
  MAXENT.path = paste(getwd(), "/models/maxent_", species.name, sep=""),
  GBM.formula = NULL, GBM.n.trees = 2001,
  GBMSTEP.gbm.x = 2:(ncol(TrainData1)), GBMSTEP.tree.complexity = 5,
  GBMSTEP.learning.rate = 0.005,
  GBMSTEP.bag.fraction = 0.5, GBMSTEP.step.size = 100,
  RF.formula = NULL, RF.ntree = 751, RF.mtry = floor(sqrt(ncol(TrainData1)-1)),

```

```

GLM.formula = NULL, GLM.family = binomial(link = "logit"),
GLMSTEP.steps = 1000, STEP.formula = NULL, GLMSTEP.scope = NULL, GLMSTEP.k = 2,
GAM.formula = NULL, GAM.family = binomial(link = "logit"),
GAMSTEP.steps = 1000, GAMSTEP.scope = NULL, GAMSTEP.pos = 1,
MGCV.formula = NULL, MGCV.select = FALSE,
MGCVFIX.formula = NULL,
EARTH.formula = NULL,
EARTH.glm = list(family = binomial(link = "logit"), maxit = maxit),
RPART.formula = NULL, RPART.xval = 50,
NNET.formula = NULL, NNET.size = 8, NNET.decay = 0.01,
FDA.formula = NULL,
SVM.formula = NULL,
SVME.formula = NULL,
MAHAL.shape = 1)

```

```

ensemble.test.gbm(x, p, a = NULL, an = 1000, excludep = FALSE, ext = NULL,
k = 5,
TrainData = NULL,
VIF = FALSE, COR = FALSE,
SINK = FALSE, PLOTS = FALSE,
species.name = "Species001",
Yweights = "BIOMOD",
layer.drops = NULL, factors = NULL,
GBMSTEP.gbm.x = 2:(ncol(TrainData.orig)),
complexity = c(3:6), learning = c(0.005, 0.002, 0.001),
GBMSTEP.bag.fraction = 0.5, GBMSTEP.step.size = 100)

```

```

ensemble.test.nnet(x = NULL, p = NULL, a = NULL, an = 1000, excludep = FALSE,
ext = NULL, k = 5,
TrainData = NULL,
VIF = FALSE, COR = FALSE,
SINK = FALSE, PLOTS = FALSE,
species.name = "Species001",
Yweights = "BIOMOD",
layer.drops = NULL, factors = NULL,
formulae.defaults = TRUE, maxit = 100,
NNET.formula = NULL,
sizes = c(2, 4, 6, 8), decays = c(0.1, 0.05, 0.01, 0.001) )

```

```

ensemble.drop1(x = NULL, p = NULL, a = NULL, an = 1000, excludep = FALSE, ext = NULL,
k = 0, pt = NULL, at = NULL,
TrainData = NULL, TestData = NULL,
VIF = FALSE, COR = FALSE,
SINK = FALSE,
species.name = "Species001",
difference = FALSE,
ENSEMBLE.best = 0, ENSEMBLE.min = 0.7, ENSEMBLE.exponent = 1,
input.weights = NULL,

```

```

MAXENT = 1, GBM = 1, GBMSTEP = 1, RF = 1, GLM = 1, GLMSTEP = 1,
GAM = 1, GAMSTEP = 1, MGCV = 1, MGCVFIX = 0, EARTH = 1,
RPART = 1, NNET = 1, FDA = 1, SVM = 1, SVME = 1,
BIOCLIM = 1, DOMAIN = 1, MAHAL = 1,
PROBIT = FALSE,
Yweights = "BIOMOD",
layer.drops = NULL, factors = NULL, dummy.vars = NULL,
maxit = 100,
MAXENT.a = NULL, MAXENT.an = 10000, MAXENT.BackData = NULL,
MAXENT.path = paste(getwd(), "/models/maxent_", species.name, sep=""),
GBM.n.trees = 2001,
GBMSTEP.tree.complexity = 5, GBMSTEP.learning.rate = 0.005,
GBMSTEP.bag.fraction = 0.5, GBMSTEP.step.size = 100,
RF.ntree = 751,
GLM.family = binomial(link = "logit"),
GLMSTEP.steps = 1000, GLMSTEP.scope = NULL, GLMSTEP.k = 2,
GAM.family = binomial(link = "logit"),
GAMSTEP.steps = 1000, GAMSTEP.scope = NULL, GAMSTEP.pos = 1,
MGCV.select = FALSE,
EARTH.glm = list(family = binomial(link = "logit"), maxit = maxit),
RPART.xval = 50,
NNET.size = 8, NNET.decay = 0.01,
MAHAL.shape = 1)

ensemble.weights(weights = c(0.9, 0.8, 0.7, 0.5),
  best = 0, min.weight = 0,
  exponent = 1, digits = 4)

ensemble.strategy(TrainData = NULL, TestData = NULL,
  verbose = FALSE,
  ENSEMBLE.best = c(4:10), ENSEMBLE.min = c(0.7),
  ENSEMBLE.exponent = c(1, 2, 4, 6, 8) )

ensemble.formulae(x, factors = NULL, dummy.vars = NULL)

```

Arguments

x	RasterStack object (stack) containing all layers that correspond to explanatory variables
p	presence points used for calibrating the suitability models, typically available in 2-column (lon, lat) dataframe; see also prepareData and extract
a	background points used for calibrating the suitability models (except for maxent), typically available in 2-column (lon, lat) dataframe; see also prepareData and extract
an	number of background points for calibration to be selected with randomPoints in case argument a is missing

excluedept	parameter that indicates (if TRUE) that presence points will be excluded from the background points; see also randomPoints
ext	an Extent object to limit the selection of background points to a sub-region of x, typically provided as c(lonmin, lonmax, latmin, latmax); see also randomPoints and extent
k	If larger than 1, the number of groups to split between calibration (k-1) and evaluation (1) data sets (for example, k = 5 results in 4/5 of presence and background points to be used for calibrating the models, and 1/5 of presence and background points to be used for evaluating the models). For <code>ensemble.test.splits</code> , <code>ensemble.test.gbm</code> and <code>ensemble.test.nnet</code> , this procedure is repeated k times (k-fold cross-validation). See also kfold .
pt	presence points used for evaluating the suitability models, available in 2-column (lon, lat) dataframe; see also prepareData and extract
at	background points used for evaluating the suitability models, available in 2-column (lon, lat) dataframe; see also prepareData and extract
TrainData	dataframe with first column 'pb' describing presence (1) and absence (0) and other columns containing explanatory variables; see also prepareData . In case that this dataframe is provided, then locations p and a are not used. For the maximum entropy model (maxent), a different dataframe is used for calibration; see parameter <code>MAXENT.TrainData</code> .
TestData	dataframe with first column 'pb' describing presence (1) and absence (0) and other columns containing explanatory variables; see also prepareData . In case that this dataframe is provided, then locations pt and at are not used. For <code>ensemble.strategy</code> , this dataframe should be a dataframe that contains predictions for various models - such dataframe can be provided by the <code>ensemble.test</code> or ensemble.raster functions.
VIF	Estimate the variance inflation factors based on a linear model calibrated on the training data (if TRUE). Only background locations will be used and the response variable 'pb' will be replaced by a random variable. See also vif .
COR	Provide information on the correlation between the numeric explanatory variables (if TRUE). See also cor .
SINK	Append the results to a text file in subfolder 'outputs' (if TRUE). The name of file is based on argument <code>species.name</code> . In case the file already exists, then results are appended. See also sink .
PLOTS	Plot the results of evaluation for the various suitability models (if TRUE). See also evaluate .
threshold.method	Method to calculate the threshold between predicted absence and presence; possibilities include <code>spec_sens</code> (highest sum of the true positive rate and the true negative rate), <code>kappa</code> (highest kappa value), <code>no_omission</code> (highest threshold that corresponds to no omission), <code>prevalence</code> (modeled prevalence is closest to observed prevalence) and <code>equal_sens_spec</code> (equal true positive rate and true negative rate). See threshold . Options specific to the BiodiversityR implementation are: <code>threshold.mean</code> (resulting in calculating the mean value of <code>spec_sens</code> , <code>equal_sens_spec</code> and <code>prevalence</code>) and <code>threshold.min</code> (resulting in calculating the minimum value of <code>spec_sens</code> , <code>equal_sens_spec</code> and <code>prevalence</code>).

threshold.sensitivity	Sensitivity value for threshold.method = 'sensitivity'. See threshold .
evaluations.keep	Keep the results of evaluations (if TRUE). See also evaluate .
models.list	list with 'old' model objects such as MAXENT or RF.
models.keep	store the details for each suitability modelling algorithm (if TRUE). (This may be particularly useful when projecting to different possible future climates.)
models.save	Save the list with model details to a file (if TRUE). The filename will be species.name with extension .models; this file will be saved in subfolder of models. When loading this file, model results will be available as ensemble.models.
species.name	Name by which the model details will be saved to a file; see also argument models.save
data.keep	Keep the data for each k-fold cross-validation run (if TRUE).
AUC.weights	If TRUE, then AUC values are used as weights for the ensemble model, scaled to sum to 1 for the ensemble model through ensemble.weights. If FALSE, then input weights are scaled to sum to 1 for the ensemble model, but AUC values are not considered.
ENSEMBLE.tune	Determine weights for the ensemble model based on AUC values through internal cross-validation procedures (if TRUE). See details.
ENSEMBLE.best	The number of individual suitability models to be used in the consensus suitability map (based on a weighted average). In case this parameter is smaller than 1 or larger than the number of positive input weights of individual models, then all individual suitability models with positive input weights are included in the consensus suitability map. In case a vector is provided, ensemble.strategy is called internally to determine weights for the ensemble model.
ENSEMBLE.min	The minimum input weight (assumed to correspond to evaluation indices of suitability models) for a model to be included in the ensemble. In case a vector is provided, function ensemble.strategy is called internally to determine weights for the ensemble model.
ENSEMBLE.exponent	Exponent applied to AUC values to convert AUC values into weights (for example, an exponent of 2 converts input weights of 0.7, 0.8 and 0.9 into $0.7^2=0.49$, $0.8^2=0.64$ and $0.9^2=0.81$). See details.
input.weights	array with numeric values for the different modelling algorithms; if NULL then values provided by parameters such as MAXENT and GBM will be used. As an alternative, the output from ensemble.test.splits can be used.
MAXENT	number: if larger than 0, then a maximum entropy model (maxent) will be fitted among ensemble
GBM	number: if larger than 0, then a boosted regression trees model (gbm) will be fitted among ensemble
GBMSTEP	number: if larger than 0, then a stepwise boosted regression trees model (gbm.step) will be fitted among ensemble
RF	number: if larger than 0, then a random forest model (randomForest) will be fitted among ensemble

GLM	number: if larger than 0, then a generalized linear model (glm) will be fitted among ensemble
GLMSTEP	number: if larger than 0, then a stepwise generalized linear model (stepAIC) will be fitted among ensemble
GAM	number: if larger than 0, then a generalized additive model (gam) will be fitted among ensemble
GAMSTEP	number: if larger than 0, then a stepwise generalized additive model (step.gam) will be fitted among ensemble
MGCV	number: if larger than 0, then a generalized additive model (gam) will be fitted among ensemble
MGCVFIX	number: if larger than 0, then a generalized additive model with fixed d.f. regression splines (gam) will be fitted among ensemble
EARTH	number: if larger than 0, then a multivariate adaptive regression spline model (earth) will be fitted among ensemble
RPART	number: if larger than 0, then a recursive partitioning and regression tree model (rpart) will be fitted among ensemble
NNET	number: if larger than 0, then an artificial neural network model (nnet) will be fitted among ensemble
FDA	number: if larger than 0, then a flexible discriminant analysis model (fda) will be fitted among ensemble
SVM	number: if larger than 0, then a support vector machine model (ksvm) will be fitted among ensemble
SVME	number: if larger than 0, then a support vector machine model (svm) will be fitted among ensemble
BIOCLIM	number: if larger than 0, then the BIOCLIM algorithm (bioclim) will be fitted among ensemble
DOMAIN	number: if larger than 0, then the DOMAIN algorithm (domain) will be fitted among ensemble
MAHAL	number: if larger than 0, then the Mahalanobis algorithm (mahal) will be fitted among ensemble
PROBIT	If TRUE, then subsequently to the fitting of the individual algorithm (e.g. maximum entropy or GAM) a generalized linear model (glm) with probit link family=binomial(link="probit") will be fitted to transform the predictions, using the previous predictions as explanatory variable. This transformation results in all model predictions to be probability estimates.
GEODIST	number: if larger than 0, then the geoDist algorithm (geoDist) will be fitted among ensemble (note that this algorithm does not use environmental layers, and is based only on the distance from presence points used to calibrate this algorithm)
Yweights	chooses how cases of presence and background (absence) are weighted; "BIOMOD" results in equal weighting of all presence and all background cases, "equal" results in equal weighting of all cases. The user can supply a vector of weights similar to the number of cases in the calibration data set.

layer.drops	vector that indicates which layers should be removed from RasterStack x. This argument is especially useful for the ensemble.drop1 function. See also addLayer .
factors	vector that indicates which variables are factors; see also prepareData
dummy.vars	vector that indicates which variables are dummy variables (influences formulae suggestions)
formulae.defaults	Suggest formulae for most of the models (if TRUE). See also ensemble.formulae .
maxit	Maximum number of iterations for some of the models. See also glm.control , gam.control , gam.control and nnet .
MAXENT.a	background points used for calibrating the maximum entropy model (maxent), typically available in 2-column (lon, lat) dataframe; see also prepareData and extract . Ignored if MAXENT.BackData is provided.
MAXENT.an	number of background points for calibration to be selected with randomPoints in case argument MAXENT.a is missing
MAXENT.BackData	dataframe containing explanatory variables for the background locations. This information will be used for calibrating the maximum entropy model (maxent). When used with the ensemble.test.splits function, the same background locations will be used for each of the cross-validation runs; this is based on the assumption that a large number (~10000) of background locations are used.
MAXENT.path	path to the directory where output files of the maximum entropy model are stored; see also maxent
GBM.formula	formula for the boosted regression trees algorithm; see also gbm
GBM.n.trees	total number of trees to fit for the boosted regression trees model; see also gbm
GBMSTEP.gbm.x	indices of column numbers with explanatory variables for stepwise boosted regression trees; see also gbm.step
GBMSTEP.tree.complexity	complexity of individual trees for stepwise boosted regression trees; see also gbm.step
GBMSTEP.learning.rate	weight applied to individual trees for stepwise boosted regression trees; see also gbm.step
GBMSTEP.bag.fraction	proportion of observations used in selecting variables for stepwise boosted regression trees; see also gbm.step
GBMSTEP.step.size	number of trees to add at each cycle for stepwise boosted regression trees (should be small enough to result in a smaller holdout deviance than the initial number of trees [50]); see also gbm.step
RF.formula	formula for random forest algorithm; see also randomForest
RF.ntree	number of trees to grow for random forest algorithm; see also randomForest
RF.mtry	number of variables randomly sampled as candidates at each split for random forest algorithm; see also randomForest

GLM.formula	formula for the generalized linear model; see also glm
GLM.family	description of the error distribution and link function for the generalized linear model; see also glm
GLMSTEP.steps	maximum number of steps to be considered for stepwise generalized linear model; see also stepAIC
STEP.formula	formula for the "starting model" to be considered for stepwise generalized linear model; see also stepAIC
GLMSTEP.scope	range of models examined in the stepwise search; see also stepAIC
GLMSTEP.k	multiple of the number of degrees of freedom used for the penalty (only $k = 2$ gives the genuine AIC); see also stepAIC
GAM.formula	formula for the generalized additive model; see also gam
GAM.family	description of the error distribution and link function for the generalized additive model; see also gam
GAMSTEP.steps	maximum number of steps to be considered in the stepwise generalized additive model; see also step.gam
GAMSTEP.scope	range of models examined in the step-wise search n the stepwise generalized additive model; see also step.gam
GAMSTEP.pos	parameter expected to be set to 1 to allow for fitting of the stepwise generalized additive model
MGCV.formula	formula for the generalized additive model; see also gam
MGCV.select	if TRUE, then the smoothing parameter estimation that is part of fitting can completely remove terms from the model; see also gam
MGCVFIX.formula	formula for the generalized additive model with fixed d.f. regression splines; see also gam (the default formulae sets "s(..., fx = TRUE, ...)"; see also s)
EARTH.formula	formula for the multivariate adaptive regression spline model; see also earth
EARTH.glm	list of arguments to pass on to glm ; see also earth
RPART.formula	formula for the recursive partitioning and regression tree model; see also rpart
RPART.xval	number of cross-validations for the recursive partitioning and regression tree model; see also rpart.control
NNET.formula	formula for the artificial neural network model; see also nnet
NNET.size	number of units in the hidden layer for the artificial neural network model; see also nnet
NNET.decay	parameter of weight decay for the artificial neural network model; see also nnet
FDA.formula	formula for the flexible discriminant analysis model; see also fda
SVM.formula	formula for the support vector machine model; see also ksvm
SVME.formula	formula for the support vector machine model; see also svm
MAHAL.shape	parameter that influences the transformation of output values of mahal . See details section.
RASTER.format	Format of the raster files that will be generated for the GEODIST model. See writeFormats and writeRaster .

complexity	vector with values of complexity of individual trees (<code>tree.complexity</code>) for boosted regression trees; see also gbm.step
learning	vector with values of weights applied to individual trees (<code>learning.rate</code>) for boosted regression trees; see also gbm.step
sizes	vector with values of number of units in the hidden layer for the artificial neural network model; see also nnet
decays	vector with values of weight decay for the artificial neural network model; see also nnet
difference	if TRUE, then AUC values of the models with all variables are subtracted from the models where one explanatory variable was excluded. After subtraction, positive values indicate that the model without the explanatory variable has a higher AUC than the model with all variables.
weights	input weights for the <code>ensemble.weights</code> function
best	The number of final weights. In case this parameter is smaller than 1 or larger than the number of positive input weights of individual models, then this parameter is ignored.
min.weight	The minimum input weight to be included in the output.
exponent	Exponent applied to AUC values to convert AUC values into weights (for example, an exponent of 2 converts input weights of 0.7, 0.8 and 0.9 into $0.7^2=0.49$, $0.8^2=0.64$ and $0.9^2=0.81$). See details.
digits	Number of number of decimal places in the output weights; see also round .
verbose	If TRUE, then provide intermediate results for <code>ensemble.strategy</code>

Details

The basic function `ensemble.test` first calibrates individual suitability models based on presence locations `p` and background locations `a`, then evaluates these suitability models based on presence locations `pt` and background locations `at`. While calibrating and testing individual models, results obtained via the [evaluate](#) function are shown in the GUI, and possibly plotted (PLOTS) or saved (`evaluations.keep`).

As an alternative to providing presence locations `p`, models can be calibrated with data provided in `TrainData`. In case that both `p` and `TrainData` are provided, then models will be calibrated with `TrainData`.

Calibration of the maximum entropy (MAXENT) algorithm is not based on background locations `a`, but based on background locations `MAXENT.a` instead. However, to compare evaluations with evaluations of other algorithms, during evaluations of the MAXENT algorithm, presence locations `p` and background locations `a` are used (and not background locations `MAXENT.a`).

As the Mahalanobis function ([mahal](#)) does not always provide values within a range of 0 - 1, the output values are rescaled by first subtracting the value of `1 - MAHAL.shape` from each prediction, followed by calculating the absolute value, followed by calculating the reciprocal value and finally multiplying this reciprocal value with `MAHAL.shape`. As this rescaling method does not estimate probabilities, inclusion in the calculation of a (weighted) average of ensemble probabilities may be problematic (the same applies to other distance-based methods).

With parameter `ENSEMBLE.best`, the subset of best models (evaluated by the individual AUC values) can be selected and only those models will be used for calculating the ensemble model (in other

words, weights for models not included in the ensemble will be set to zero). It is possible to further increase the contribution to the ensemble model for models with higher AUC values through parameter `ENSEMBLE.exponent`. With `ENSEMBLE.exponent = 2`, AUC values of 0.7, 0.8 and 0.9 are converted into weights of $0.7^2=0.49$, $0.8^2=0.64$ and $0.9^2=0.81$). With `ENSEMBLE.exponent = 4`, AUC values of 0.7, 0.8 and 0.9 are converted into weights of $0.7^4=0.2401$, $0.8^4=0.4096$ and $0.9^4=0.6561$).

`ENSEMBLE.tune` will result in an internal procedure whereby the best selection of parameter values for `ENSEMBLE.min`, `ENSEMBLE.best` or `ENSEMBLE.exponent` can be identified. Through a factorial procedure, the ensemble model with best AUC for a specific combination of parameter values is identified. The procedure also provides the weights that correspond to the best ensemble.

Function `ensemble.test.splits` splits the presence and background locations in a user-defined (k) number of subsets (i.e. k-fold cross-validation), then sequentially calibrates individual suitability models with (k-1) combined subsets and evaluates those with the remaining one subset, whereby each subset is used once for evaluation in the user-defined number (k) of runs. For example, `k = 5` results in splitting the locations in 5 subsets, then using one of these subsets in turn for evaluations (see also `kfold`). Note that for the maximum entropy (MAXENT) algorithm, the same background data will be used in each cross-validation run (this is based on the assumption that a large number (~10000) of background locations are used).

Among the output from function `ensemble.test.splits` are suggested weights for an ensemble model (`output.weights` and `output.weights.AUC`), and information on the respective AUC values of the ensemble model with the suggested weights for each of the (k) subsets. Suggested weights `output.weights` are calculated as the average of the weights of the different algorithms (submodels) of the k ensembles. Suggested weights `output.weights.AUC` are calculated as the average of the AUC of the different algorithms of the for the k runs.

Function `ensemble.test.gbm` allows to test various combinations of parameters `tree.complexity` and `learning.rate` for the `gbm.step` model.

Function `ensemble.test.nnet` allows to test various combinations of parameters `size` and `decay` for the `nnet` model.

Function `ensemble.drop1` allows to test the effects of leaving out each of the explanatory variables, and comparing these results with the "full" model. Note that option of `difference = TRUE` may result in positive values, indicating that the model without the explanatory variable having larger AUC than the "full" model. A procedure is included to estimate the deviance of a model based on the fitted values, using $-2 * (\sum(x * \log(x)) + \sum((1-x) * \log(1-x)))$ where x is a vector of the fitted values for a respective model. (It was checked that this procedure results in similar deviance estimates for the null and 'full' models for `glm`, but note that it is not certain whether deviance can be calculated in a similar way for other submodels.)

Function `ensemble.formulae` provides suggestions for formulae that can be used for `ensemble.test` and `ensemble.raster`. This function is always used internally by the `ensemble.drop1` function.

The `ensemble.weights` function is used internally by the `ensemble.test` and `ensemble.raster` functions, using the input weights for the different suitability modelling algorithms. Ties between input weights result in the same output weights.

The `ensemble.strategy` function is used internally by the `ensemble.test` and `ensemble.raster` functions, using the train and test data sets with predictions of the different suitability modelling algorithms and different combinations of parameters `ENSEMBLE.best`, `ENSEMBLE.min` and `ENSEMBLE.exponent`. The final ensemble model is based on the parameters that generate the best AUC.

Value

Function `ensemble.test` (potentially) returns a list with results from evaluations (via [evaluate](#)) of calibration and test runs of individual suitability models.

Function `ensemble.test.splits` returns a matrix with, for each individual suitability model, the AUC of each run and the average AUC over the runs. Models are sorted by the average AUC. The average AUC for each model can be used as input weights for the [ensemble.raster](#) function.

Functions `ensemble.test.gbm` and `ensemble.test.nnet` return a matrix with, for each combination of model parameters, the AUC of each run and the average AUC. Models are sorted by the average AUC.

Author(s)

Roeland Kindt (World Agroforestry Centre)

References

Buisson L, Thuiller W, Casajus N, Lek S and Grenouillet G. 2010. Uncertainty in ensemble forecasting of species distribution. *Global Change Biology* 16: 1145-1157

See Also

[ensemble.raster](#)

Examples

```
## Not run:
# based on examples in the dismo package

# get predictor variables
library(dismo)
predictor.files <- list.files(path=paste(system.file(package="dismo"), '/ex', sep=''),
                             pattern='grd', full.names=TRUE)
predictors <- stack(predictor.files)
# subset based on Variance Inflation Factors
predictors <- subset(predictors, subset=c("bio5", "bio6",
    "bio16", "bio17", "biome"))
predictors
predictors@title <- "base"

# presence points
presence_file <- paste(system.file(package="dismo"), '/ex/bradypus.csv', sep='')
pres <- read.table(presence_file, header=TRUE, sep=',')[,-1]

# the kfold function randomly assigns data to groups;
# groups are used as calibration (1/5) and training (4/5) data
group <- kfold(pres, 5)
pres_train <- pres[group != 1, ]
pres_test <- pres[group == 1, ]

# choose background points
```

```

ext <- extent(-90, -32, -33, 23)
background <- randomPoints(predictors, n=1000, ext=ext, extf=1.00)
colnames(background)=c('lon', 'lat')
groupa <- kfold(background, 5)
backg_train <- background[groupa != 1, ]
backg_test <- background[groupa == 1, ]

# formulae for random forest and generalized linear model
# compare with: ensemble.formulae(predictors, factors=c("biome"))

rfformula <- as.formula(pb ~ bio5+bio6+bio16+bio17)

glmformula <- as.formula(pb ~ bio5 + I(bio5^2) + I(bio5^3) +
  bio6 + I(bio6^2) + I(bio6^3) + bio16 + I(bio16^2) + I(bio16^3) +
  bio17 + I(bio17^2) + I(bio17^3) )

# fit four ensemble models (RF, GLM, BIOCLIM, DOMAIN)
ensemble.nofactors <- ensemble.test(x=predictors, p=pres_train, a=backg_train,
  pt=pres_test, at=backg_test,
  species.name="Bradypus",
  MAXENT=0, GBM=0, GBMSTEP=0, RF=1, GLM=1, GLMSTEP=0, GAM=0,
  GAMSTEP=0, MGCV=0, MGCVFIX=0, EARTH=0, RPART=0, NNET=0, FDA=0,
  SVM=0, SVME=0, BIOCLIM=1, DOMAIN=1, MAHAL=0,
  Yweights="BIOMOD", factors="biome",
  PLOTS=FALSE, evaluations.keep=TRUE,
  RF.formula=rfformula,
  GLM.formula=glmformula)

# fit four ensemble models (RF, GLM, BIOCLIM, DOMAIN) using default formulae
# variable 'biome' is not included as explanatory variable
# results are provided in a file in the 'outputs' subfolder of the working
# directory
ensemble.nofactors <- ensemble.test(x=predictors,
  p=pres_train, a=backg_train,
  pt=pres_test, at=backg_test,
  layer.drops="biome",
  species.name="Bradypus",
  SINK=TRUE,
  MAXENT=0, GBM=0, GBMSTEP=0, RF=1, GLM=1, GLMSTEP=0, GAM=0,
  GAMSTEP=0, MGCV=0, MGCVFIX=0, EARTH=0, RPART=0, NNET=0, FDA=0,
  SVM=0, SVME=0, BIOCLIM=1, DOMAIN=1, MAHAL=0,
  Yweights="BIOMOD", factors="biome",
  PLOTS=FALSE, evaluations.keep=TRUE,
  formulae.defaults=TRUE)

# after fitting the individual algorithms (submodels),
# transform predictions with a probit link.
ensemble.nofactors <- ensemble.test(x=predictors,
  p=pres_train, a=backg_train,
  pt=pres_test, at=backg_test,
  layer.drops="biome",
  species.name="Bradypus",
  SINK=TRUE,

```

```

ENSEMBLE.min=0.6,
MAXENT=0, GBM=0, GBMSTEP=0, RF=1, GLM=1, GLMSTEP=0, GAM=0,
GAMSTEP=0, MGCV=0, MGCVFIX=0, EARTH=0, RPART=0, NNET=0, FDA=0,
SVM=0, SVME=0, BIOCLIM=1, DOMAIN=1, MAHAL=0,
PROBIT=TRUE,
Yweights="BIOMOD", factors="biome",
PLOTS=FALSE, evaluations.keep=TRUE,
formulae.defaults=TRUE)

# instead of providing presence and background locations, provide data.frames
# because 'biome' is a factor, RasterStack and extent need to be provided
# to check for levels in the Training and Testing data set
TrainData1 <- prepareData(x=predictors, p=pres_train, b=backg_train,
  factors=c("biome"), xy=FALSE)
TestData1 <- prepareData(x=predictors, p=pres_test, b=backg_test,
  factors=c("biome"), xy=FALSE)
ensemble.factors1 <- ensemble.test(x=predictors, ext=ext,
  TrainData=TrainData1, TestData=TestData1,
  p=pres_train, a=backg_train,
  pt=pres_test, at=backg_test,
  species.name="Bradypus",
  SINK=TRUE,
  MAXENT=1, GBM=1, GBMSTEP=0, RF=1, GLM=1, GLMSTEP=1, GAM=1,
  GAMSTEP=1, MGCV=1, MGCVFIX=1, EARTH=1, RPART=1, NNET=1, FDA=1,
  SVM=1, SVME=1, BIOCLIM=1, DOMAIN=1, MAHAL=0,
  Yweights="BIOMOD", factors="biome",
  PLOTS=FALSE, evaluations.keep=TRUE)

# compare different methods of calculating ensembles
ensemble.factors2 <- ensemble.test(x=predictors, ext=ext,
  TrainData=TrainData1, TestData=TestData1,
  p=pres_train, a=backg_train,
  pt=pres_test, at=backg_test,
  species.name="Bradypus",
  SINK=TRUE,
  MAXENT=1, GBM=1, GBMSTEP=0, RF=1, GLM=1, GLMSTEP=1, GAM=1,
  GAMSTEP=1, MGCV=1, MGCVFIX=1, EARTH=1, RPART=1, NNET=1, FDA=1,
  SVM=1, SVME=1, BIOCLIM=1, DOMAIN=1, MAHAL=0,
  ENSEMBLE.best=c(4:10), ENSEMBLE.exponent=c(1, 2, 4, 6, 8),
  Yweights="BIOMOD", factors="biome",
  PLOTS=FALSE, evaluations.keep=TRUE)

# test performance of different suitability models
# data are split in 4 subsets, each used once for evaluation
ensemble.nofactors2 <- ensemble.test.splits(x=predictors, ext=ext,
  p=pres, a=background, k=4,
  layer.drops=c("biome"),
  species.name="Bradypus",
  SINK=TRUE,
  MAXENT=1, GBM=1, GBMSTEP=0, RF=1, GLM=1, GLMSTEP=1, GAM=1,
  GAMSTEP=1, MGCV=1, MGCVFIX=1, EARTH=1, RPART=1, NNET=1, FDA=1,
  SVM=1, SVME=1, BIOCLIM=1, DOMAIN=1, MAHAL=0,

```

```

ENSEMBLE.best=0, ENSEMBLE.exponent=c(1, 2, 4, 6, 8),
ENSEMBLE.min=0.7,
Yweights="BIOMOD", factors="biome",
PLOTS=FALSE, formulae.defaults=TRUE,
GBMSTEP.learning.rate=0.002)
ensemble.nofactors2

# test the result of leaving out one of the variables from the model
# note that positive differences indicate that the model without the variable
# has higher AUC than the full model
ensemble.variables <- ensemble.drop1(x=predictors, ext=ext,
  p=pres, a=background, k=5,
  layer.drops=c("bio6", "bio1", "bio12"),
  species.name="Bradypus",
  SINK=TRUE,
  difference=TRUE,
  VIF=TRUE,
  MAXENT=0, GBM=1, GBMSTEP=0, RF=1, GLM=1, GLMSTEP=1, GAM=1,
  GAMSTEP=1, MGCV=1, MGCVFIX=1, EARTH=1, RPART=1, NNET=1, FDA=1,
  SVM=1, SVME=1, BIOCLIM=0, DOMAIN=0, MAHAL=0,
  ENSEMBLE.best=0, ENSEMBLE.exponent=c(1, 2, 4, 6, 8),
  ENSEMBLE.min=0.7,
  Yweights="BIOMOD", factors="biome",
  GBMSTEP.learning.rate=0.002)
ensemble.variables

## End(Not run)

```

evaluation.strip.data *Evaluation strips for ensemble suitability mapping*

Description

These functions provide a dataframe which can subsequently be used to evaluate the relationship between environmental variables and the fitted probability of occurrence of individual or ensemble suitability modelling algorithms. The biomod2 package provides an alternative implementation of this approach (`response.plot2`).

Usage

```

evaluation.strip.data(xn = NULL, ext = NULL,
  models.list = NULL,
  input.weights = models.list$output.weights,
  vars = models.list$vars, factors = models.list$factors,
  dummy.vars = models.list$dummy.vars,
  steps=50
)

```

```

evaluation.strip.plot(data,
  modelnames = c("MAXENT", "GBM", "GBMSTEP", "RF", "GLM", "GLMSTEP", "GAM",
    "GAMSTEP", "MGCV", "MGCVFIX", "EARTH", "RPART", "NNET", "FDA",
    "SVM", "SVME", "BIOCLIM", "DOMAIN", "MAHAL"),
  variable = NULL, model = NULL, ...
)

```

Arguments

xn	RasterStack object (stack) containing all layers that correspond to explanatory variables of an ensemble calibrated earlier with ensemble.test . See also predict .
ext	an Extent object to limit the prediction to a sub-region of xn and the selection of background points to a sub-region of x, typically provided as c(lonmin, lonmax, latmin, latmax); see also predict , randomPoints and extent
models.list	list with 'old' model objects such as MAXENT or RF.
input.weights	array with numeric values for the different modelling algorithms; if NULL then values provided by parameters such as MAXENT and GBM will be used. As an alternative, the output from ensemble.test.splits can be used.
vars	Vector that indicates which variables should be included as columns in the data frame. Only variables that correspond to layers of the rasterStack will be included.
factors	vector that indicates which variables are factors; see also prepareData and ensemble.formulae
dummy.vars	vector that indicates which variables are dummy variables (coded 0 or 1 to indicate presence of specific level of a categorical variable; see also ensemble.formulae
steps	number of steps within the range of a continuous explanatory variable
data	data set with ranges of environmental variables and fitted suitability models, typically returned by evaluation.strip.data
modelnames	abbreviated names of the individual suitability models that are fitted. Should correspond to the names used in ensemble.grd .
variable	focal explanatory variable for plots with evaluation strips
model	focal model for plots with evaluation strips
...	Other arguments passed to plot

Details

These functions are mainly intended to be used internally by the [ensemble.raster](#) function.

[evaluation.strip.data](#) creates a data frame with variables (columns) corresponding to the environmental variables encountered in the RasterStack object (x) and the suitability modelling approaches that were defined. The variable of [focal.var](#) is an index of the variable for which values are ranged. The variable of [categorical](#) is an index for categorical (factor) variables.

A continuous (numeric) variable is ranged between its minimum and maximum values in the number of steps defined by argument `steps`. When a continuous variable is not the focal variable, then the average (`mean`) is used.

A categorical (factor) variable is ranged for all the encountered levels (`levels`) for this variable. When a categorical variable is not the focal variable, then the most frequent level is used.

Value

function `evaluation.strip.data` creates a data frame, function `codeevaluation.strip.data` allows for plotting.

Author(s)

Roeland Kindt (World Agroforestry Centre)

References

Elith J, Ferrier S, Huettmann F & Leathwick J. 2005. The evaluation strip: A new and robust method for plotting predicted responses from species distribution models. *Ecological Modelling* 186: 280-289

See Also

[ensemble.raster](#)

Examples

```
## Not run:

# get predictor variables
library(dismo)
predictor.files <- list.files(path=paste(system.file(package="dismo"), '/ex', sep=''),
  pattern='grd', full.names=TRUE)
predictors <- stack(predictor.files)
# subset based on Variance Inflation Factors
predictors <- subset(predictors, subset=c("bio5", "bio6",
  "bio16", "bio17", "biome"))
predictors
predictors@title <- "base"

# presence points
presence_file <- paste(system.file(package="dismo"), '/ex/bradypus.csv', sep='')
pres <- read.table(presence_file, header=TRUE, sep=',')[,-1]

# the kfold function randomly assigns data to groups;
# groups are used as calibration (1/5) and training (4/5) data
group <- kfold(pres, 5)
pres_train <- pres[group != 1, ]
pres_test <- pres[group == 1, ]

# choose background points
```

```

ext <- extent(-90, -32, -33, 23)
background <- randomPoints(predictors, n=1000, ext=ext, extf=1.00)
colnames(background)=c('lon', 'lat')
groupa <- kfold(background, 5)
backg_train <- background[groupa != 1, ]
backg_test <- background[groupa == 1, ]

# calibrate the models
ensemble.calibrate <- ensemble.test(x=predictors, ext=ext,
  p=pres_train, a=backg_train,
  pt=pres_test, at=backg_test,
  ENSEMBLE.min=0.6,
  MAXENT=1, GBM=1, GBMSTEP=1, RF=1, GLM=1, GLMSTEP=1, GAM=1,
  GAMSTEP=1, MGCV=1, MGCVFIX=1, EARTH=1, RPART=1, NNET=1, FDA=1,
  SVM=1, SVME=1, BIOCLIM=1, DOMAIN=1, MAHAL=1,
  Yweights="BIOMOD", factors="biome",
  PLOTS=FALSE, models.keep=TRUE)

# obtain data for plotting the evaluation strip
strip.data <- evaluation.strip.data(xn=predictors, ext=ext,
  models.list=ensemble.calibrate$models)

# create graphs
evaluation.strip.plot(data=strip.data, variable="bio5", type="o", col="red")
evaluation.strip.plot(data=strip.data, model="ENSEMBLE", type="o", col="red")

## End(Not run)

```

faramaea

Faramaea occidentalis abundance in Panama

Description

This dataset describes the abundance (number of trees with diameter at breast height equal or larger than 10 cm) of the tree species *Faramaea occidentalis* as observed in a 1-ha quadrat survey from the Barro Colorado Island of Panama. For each quadrat, some environmental characteristics are also provided.

Usage

```
data(faramaea)
```

Format

A data frame with 45 observations on the following 8 variables.

UTM.EW a numeric vector

UTM.NS a numeric vector

Precipitation a numeric vector
Elevation a numeric vector
Age a numeric vector
Age.cat a factor with levels c1 c2 c3
Geology a factor with levels pT Tb Tbo Tc Tcm Tgo Tl
Faramea.occidentalis a numeric vector

Details

Although the original survey documented tree species composition of all 1-ha subplots of larger (over 1 ha) sample plot, only the first (and sometimes the last) quadrats of the larger plots were included. This selection was made to avoid that larger sample plots dominated the analysis. This selection of sites is therefore different from the selection of the 50 1-ha quadrats of the largest sample plot of the same survey ([BCI](#) and [BCI.env](#))

This dataset is the main dataset used for the examples provided in chapters 6 and 7 of the Tree Diversity Analysis manual (Kindt & Coe, 2005).

Source

<http://www.sciencemag.org/cgi/content/full/295/5555/666/DC1>

References

Pyke CR, Condit R, Aguilar S and Lao S. (2001). Floristic composition across a climatic gradient in a neotropical lowland forest. *Journal of Vegetation Science* 12: 553-566.

Condit, R, Pitman, N, Leigh, E.G., Chave, J., Terborgh, J., Foster, R.B., Nunez, P., Aguilar, S., Valencia, R., Villa, G., Muller-Landau, H.C., Losos, E. & Hubbell, S.P. (2002). Beta-diversity in tropical forest trees. *Science* 295: 666-669.

Kindt, R. & Coe, R. (2005) Tree diversity analysis: A manual and software for common statistical methods for ecological and biodiversity studies.

<http://www.worldagroforestry.org/resources/databases/tree-diversity-analysis>

Examples

```
data(faramea)
```

Description

This data set contains information on the number of stems (individuals) and basal areas for 34 vegetation plots inventoried in February 1997 in Lothlorien forest, 37 vegetation plots inventoried in February 1996 in May Creek Forest and 36 vegetation plots inventoried in May 1995 in Yellowwood State Forest. All three sites are in Indiana, USA. Data were gathered through IFRI inventory protocols to record any tree, palm and woody climber with diameter at breast height greater than or equal to 10 cm in 10-m radius circular plots; only tree species data were kept in the example data sets (IFRI research instruments and IFRI manual section P: Forest Plot Form, section D1: Tree, Palm and Woody Climber Information).

Usage

```
data(ifri)
```

Format

A data frame with 486 observations on the following 5 variables.

forest a factor with 3 levels: "LOT" (Lothlorien forest), "MCF" (May Creek Forest) and "YSF" (Yellowwood State Forest)

plotID a factor with 107 levels providing an identification code for a 314.16 square metres (10 m radius) vegetation plot

species a factor with 50 levels providing an 8 character code for a tree species

count a numeric vector providing the number of stems (individuals) for each species in each vegetation plot

basal a numeric vector providing the basal area (calculated from the diameter at breast height) in square cm for each species in each vegetation plot

Source

IFRI (2014) Data from the International Forestry Resources and Institutions (IFRI) research network. <http://www.ifriresearch.net>

Examples

```
data(ifri)
```

importancevalue

Importance Value

Description

Calculates the importance values of tree species based on frequency (calculated from number of plots), density (calculated from number of individuals) and dominance (calculated from basal area). See details.

Usage

```
importancevalue(x, site="plotID", species="species",
               count="count", basal="basal",
               factor="forest", level="")

importancevalue.comp(x, site="plotID", species="species",
                   count="count", basal="basal",
                   factor="forest")
```

Arguments

x	data frame with information on plot identities, species identities, number of individuals and basal areas
site	factor variable providing the identities of survey plots
species	factor variable providing the identities of tree species
count	number of individuals for each tree species in each survey plot
basal	basal area for each tree species in each survey plot
factor	factor variable used to define subsets (typically different forest reserves)
level	level of the factor variable used to create a subset from the original data

Details

The importance value is calculated as the sum from (i) the relative frequency; (ii) the relative density; and (iii) the relative dominance. The importance value ranges between 0 and 300.

Frequency is calculated as the number of plots where a species is observed divided by the total number of survey plots. Relative frequency is calculated by dividing the frequency by the sum of the frequencies of all species, multiplied by 100 (to obtain a percentage).

Density is calculated as the total number of individuals of a species. Relative density is calculated by dividing the density by the sum of the densities of all species, multiplied by 100 (to obtain a percentage).

Dominance is calculated as the total basal area of a species. Relative dominance is calculated by dividing the dominance by the sum of the dominance of all species, multiplied by 100 (to obtain a percentage).

Functions `importancevalue.comp` applies function `importancevalue` to all available levels of a factor variable.

Value

Provides information on the importance value for all tree species

Author(s)

Roeland Kindt (World Agroforestry Centre), Peter Newton (University of Michigan)

References

Curtis, J.T. & McIntosh, R. P. (1951) An Upland Forest Continuum in the Prairie-Forest Border Region of Wisconsin. *Ecology* 32: 476-496.

Kent, M. (2011) *Vegetation Description and Data Analysis: A Practical Approach*. Second edition. 428 pages.

See Also

[ifri](#)

Examples

```
data(ifri)
importancevalue(ifri, site='plotID', species='species', count='count',
  basal='basal', factor='forest', level='YSF')
importancevalue.comp(ifri, site='plotID', species='species', count='count',
  basal='basal', factor='forest')

# When all survey plots are the same size, importance value
# is not affected. Counts and basal areas now calculated per square metre
ifri$count <- ifri$count/314.16
ifri$basal <- ifri$basal/314.16

importancevalue(ifri, site='plotID', species='species', count='count',
  basal='basal', factor='forest', level='YSF')
importancevalue.comp(ifri, site='plotID', species='species', count='count',
  basal='basal', factor='forest')
```

loaded.citations

Give Citation Information for all Loaded Packages

Description

This function provides citation information for all loaded packages.

Usage

```
loaded.citations()
```

Details

The function checks for the loaded packages via [.packages](#). Citation information is provided for the base package and for all the non-standard packages via [citation](#).

Value

The function provides a list of all loaded packages and the relevant citation information.

Author(s)

Roeland Kindt (World Agroforestry Centre)

makecommunitydataset *Make a Community Dataset from a Stacked Dataset*

Description

Makes a community data set from a stacked dataset (with separate variables for the site identities, the species identities and the abundance).

Usage

```
makecommunitydataset(x, row, column, value, factor="", level="", drop=F)
```

Arguments

x	Data frame.
row	Name of the categorical variable for the rows of the crosstabulation (typically indicating sites)
column	Name of the categorical variable for the columns of the crosstabulation (typically indicating species)
value	Name of numerical variable for the cells of the crosstabulation (typically indicating abundance). The cells provide the sum of all values in the data frame.
factor	Name of the variable to calculate a subset of the data frame.
level	Value of the subset of the factor variable to calculate a subset of the data frame.
drop	Drop rows without species (species with total abundance of zero are always dropped)

Details

This function calculates a cross-tabulation from a data frame, summing up all the values of the numerical variable identified as variable for the cell values. If factor="", then no subset is calculated from the data frame in the first step.

Value

The function provides a community dataset from another data frame.

Author(s)

Roeland Kindt (World Agroforestry Centre)

References

Kindt, R. & Coe, R. (2005) Tree diversity analysis: A manual and software for common statistical methods for ecological and biodiversity studies.

<http://www.worldagroforestry.org/resources/databases/tree-diversity-analysis>

Examples

```
library(vegan)
data(dune.env)
makecommunitydataset(dune.env, row='Manure', column='Use', value='A1')
makecommunitydataset(dune.env, row='Manure', column='Use', value='A1',
  factor='Management', level='NM')
```

multiconstrained	<i>Pairwise Comparisons for All Levels of a Categorical Variable by RDA, CCA or Capscale</i>
------------------	--

Description

This function implements pairwise comparisons for categorical variable through `capscale`, `cca` or `rda` followed by `anova.cca`. The function simply repeats constrained ordination analysis by selecting subsets of data that correspond to two factor levels.

Usage

```
multiconstrained(method="capscale", formula, data, distance = "bray"
  , comm = NULL, add = FALSE, multcomp="", contrast=0, ...)
```

Arguments

method	Method for constrained ordination analysis; one of "rda", "cca" or "capscale".
formula	Model formula as in <code>capscale</code> , <code>cca</code> or <code>rda</code> . The LHS can be a community data matrix or a distance matrix for <code>capscale</code> .
data	Data frame containing the variables on the right hand side of the model formula as in <code>capscale</code> , <code>cca</code> or <code>rda</code> .
distance	Dissimilarity (or distance) index in <code>vegdist</code> used if the LHS of the formula is a data frame instead of dissimilarity matrix; used only with function <code>vegdist</code> and partial match to "manhattan", "euclidean", "canberra", "bray", "kulczynski", "jaccard", "gower", "morisita", "horn" or "mountford". This argument is only used for <code>capscale</code> in case that the LHS of the formula is a community matrix.
comm	Community data frame which will be used for finding species scores when the LHS of the formula was a dissimilarity matrix as only allowed for <code>capscale</code> . This is not used if the LHS is a data frame.
add	Logical indicating if an additive constant should be computed, and added to the non-diagonal dissimilarities such that all eigenvalues are non-negative in underlying Principal Co-ordinates Analysis; only applicable in <code>capscale</code> .

multicomp	Categorical variable used to construct the contrasts from. In case that this variable is missing, then the first explanatory variable of the formula will be used.
contrast	Return the ordination results for the particular contrast indicated by this number (e.g. with 5 levels, one can choose in between contrast 1-10). In case=0, then the first row of the <code>anova.cca</code> results for all contrasts is provided.
...	Other parameters passed to <code>anova.cca</code> .

Details

This function provides a simple expansion of `capscale`, `cca` and `rda` by conducting the analysis for subsets of the community and environmental datasets that only contain two levels of a categorical variable.

When the choice is made to return results from all contrasts (`contrast=0`), then the first row of the `anova.cca` tables for each contrast are provided. It is therefore possible to compare differences in results by modifying the "by" argument of this function (i.e. obtain the total of explained variance, the variance explained on the first axis or the variance explained by the variable alone).

When the choice is made to return results from a particular contrast (`contrast>0`), then the ordination result is returned and two new datasets ("newcommunity" and "newenvdata") are created that only contain data for the two selected contrasts.

Value

The function returns an ANOVA table that contains the first rows of the ANOVA tables obtained for all possible combinations of levels of the first variable. Alternatively, it returns an ordination result for the selected contrast and creates two new datasets ("newcommunity" and "newenvdata")

Author(s)

Roeland Kindt (World Agroforestry Centre)

References

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

Anderson, M.J. & Willis, T.J. (2003). Canonical analysis of principal coordinates: a useful method of constrained ordination for ecology. *Ecology* 84: 511-525.

Examples

```
## Not run:
library(vegan)
library(MASS)
data(dune)
data(dune.env)
multiconstrained(method="capscale", dune~Management, data=dune.env,
  distance="bray", add=TRUE)
multiconstrained(method="capscale", dune~Management+Condition(A1),
  data=dune.env, distance="bray", add=TRUE, contrast=3)
```

```
## End(Not run)
```

nested.anova.dbrda	<i>Nested Analysis of Variance via Distance-based Redundancy Analysis or Non-parametric Multivariate Analysis of Variance</i>
--------------------	---

Description

The functions provide nested analysis of variance for a two-level hierarchical model. The functions are implemented by estimating the correct F-ratio for the main and nested factors (assuming the nested factor is random) and using the recommended permutation procedures to test the significance of these F-ratios. F-ratios are estimated from variance estimates that are provided by distance-based redundancy analysis ([capscale](#)) or non-parametric multivariate analysis of variance ([adonis](#)).

Usage

```
nested.anova.dbrda(formula, data, method="euc", add=FALSE,
  permutations=100, warnings=FALSE)
nested.npmanova(formula, data, method="euc", permutations=100, warnings=FALSE)
```

Arguments

formula	Formula with a community data frame (with sites as rows, species as columns and species abundance as cell values) or (for nested.anova.dbrda only) distance matrix on the left-hand side and two categorical variables on the right-hand side (with the second variable assumed to be nested within the first).
data	Environmental data set.
method	Method for calculating ecological distance with function vegdist : partial match to "manhattan", "euclidean", "canberra", "bray", "kulczynski", "jaccard", "gower", "morisita", "horn" or "mountford". This argument is ignored in case that the left-hand side of the formula already is a distance matrix.
add	Should a constant be added to the off-diagonal elements of the distance-matrix (TRUE) or not.
permutations	The number of permutations for significance testing.
warnings	Should warnings be suppressed (TRUE) or not.

Details

The functions provide two alternative procedures for multivariate analysis of variance on the basis of any distance measure. Function nested.anova.dbrda proceeds via [capscale](#), whereas nested.npmanova proceeds via [adonis](#). Both methods are complementary to each other as nested.npmanova always provides correct F-ratios and estimations of significance, whereas nested.anova.dbrda does not provide correct F-ratios and estimations of significance when negative eigenvalues are encountered or constants are added to the distance matrix, but always provides an ordination diagram.

The F-ratio for the main factor is estimated as the mean square of the main factor divided by the mean square of the nested factor. The significance of the F-ratio of the main factor is tested by

permuting entire blocks belonging to levels of the nested factor. The significance of the F-ratio of the nested factor is tested by permuting sample units within strata defined by levels of the main factor.

Value

The functions provide an ANOVA table.

Author(s)

Roeland Kindt (World Agroforestry Centre)

References

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

Anderson, M.J. (2001). A new method for non-parametric multivariate analysis of variance. *Austral Ecology*, 26: 32-46.

McArdle, B.H. and M.J. Anderson. (2001). Fitting multivariate models to community data: A comment on distance-based redundancy analysis. *Ecology*, 82: 290-297.

Examples

```
## Not run:
library(vegan)
data(warcom)
data(warenv)
# use larger number of permutations for real studies
nested.npmanova(warcom~rift.valley+popshort, data=warenv, method="jac",
  permutations=5)
nested.anova.dbrda(warcom~rift.valley+popshort, data=warenv, method="jac",
  permutations=5)

## End(Not run)
```

NMSrandom

Calculate the NMS Result with the Smallest Stress from Various Random Starts

Description

This function provides a simplified version of the method of calculating NMS results implemented by the function `metaMDS` ([vegan](#)).

Usage

```
NMSrandom(x, perm=100, k=2, stressresult=F, method="isoMDS")
```

Arguments

x	Distance matrix.
perm	Number of permutations to select the configuration with the lowest stress.
k	Number of dimensions for the non metric scaling result; passed to isoMDS or sammon .
stressresult	Provide the calculated stress for each permutation.
method	Method for calculating the NMS: isoMDS or sammon .

Details

This function is an easier method of calculating the best NMS configuration after various random starts than implemented in the [metaMDS](#) function (**vegan**). The function uses a distance matrix (as calculated for example by function [vegdist](#) from a community data set) and calculates random starting positions by function [initMDS](#) (**vegan**) analogous to [metaMDS](#).

Value

The function returns the NMS ordination result with the lowest stress (calculated by [isoMDS](#) or [sammon](#)), or the stress of each NMS ordination.

Author(s)

Roeland Kindt (World Agroforestry Centre)

References

Kindt, R. & Coe, R. (2005) Tree diversity analysis: A manual and software for common statistical methods for ecological and biodiversity studies.

<http://www.worldagroforestry.org/resources/databases/tree-diversity-analysis>

Examples

```
library(vegan)
library(MASS)
data(dune)
distmatrix <- vegdist(dune)
Ordination.model1 <- NMSrandom(distmatrix,perm=100,k=2)
Ordination.model1 <- add.spec.scores(Ordination.model1,dune,
  method='wa.scores')
Ordination.model1
```

nnetrandom	<i>Calculate the NNET Result with the Smallest Value from Various Random Starts</i>
------------	---

Description

This function provides the best solution from various calls to the [nnet](#) feed-forward artificial neural networks function (**nnet**).

Usage

```
nnetrandom(formula,data,tries=10,leave.one.out=F,...)
```

Arguments

formula	Formula as passed to nnet .
data	Data as passed to nnet .
tries	Number of calls to nnet to obtain the best solution.
leave.one.out	Calculate leave-one-out predictions.
...	Other arguments passed to nnet .

Details

This function makes various calls to [nnet](#). If desired by the user, leave-one-out statistics are provided that report the prediction if one particular sample unit was not used for iterating the networks.

Value

The function returns the same components as [nnet](#), but adds the following components:

range	Summary of the observed "values".
tries	Number of different attempts to iterate an ANN.
CV	Predicted class when not using the respective sample unit for iterating ANN.
successful	Test whether leave-one-out statistics provided the same class as the original class.

Author(s)

Roeland Kindt (World Agroforestry Centre)

Examples

```
## Not run:
data(faramea)
faramea <- na.omit(faramea)
faramea$presence <- as.numeric(faramea$Faramea occidentalis > 0)
attach(faramea)
library(nnet)
result <- nnetrandom(presence ~ Elevation, data=faramea, size=2,
  skip=FALSE, entropy=TRUE, trace=FALSE, maxit=1000, tries=100,
  leave.one.out=FALSE)
summary(result)
result$fitted.values
result$value
result2 <- nnetrandom(presence ~ Elevation, data=faramea, size=2,
  skip=FALSE, entropy=TRUE, trace=FALSE, maxit=1000, tries=50,
  leave.one.out=TRUE)
result2$range
result2$CV
result2$successful

## End(Not run)
```

ordicoeno

Coenoclines for an Ordination Axis

Description

A graph is produced that summarizes (through GAM as implemented by [gam](#)) how the abundance of all species of the community data set change along an ordination axis (based on the position of sites along the axis and the information from the community data set).

Usage

```
ordicoeno(x, ordiplot, axis = 1, legend = FALSE, cex = 0.8, ncol = 4, ...)
```

Arguments

x	Community data frame with sites as rows, species as columns and species abundance as cell values.
ordiplot	Ordination plot created by ordiplot .
axis	Axis of the ordination graph (1: horizontal, 2: vertical).
legend	if TRUE, then add a legend to the plot.
cex	the amount by which plotting text and symbols should be magnified relative to the default; see also par
ncol	the number of columns in which to set the legend items; see also legend
...	Other arguments passed to functions plot and points .

Details

This functions investigates the relationship between the species vectors and the position of sites on an ordination axis. A GAM ([gam](#)) investigates the relationship by using the species abundances of each species as response variable, and the site position as the explanatory variable. The graph shows how the abundance of each species changes over the gradient of the ordination axis.

Value

The function plots coenoclines and provides the expected degrees of freedom (complexity of the relationship) estimated for each species by GAM.

Author(s)

Roeland Kindt (World Agroforestry Centre)

References

Kindt, R. & Coe, R. (2005) Tree diversity analysis: A manual and software for common statistical methods for ecological and biodiversity studies.

<http://www.worldagroforestry.org/resources/databases/tree-diversity-analysis>

Examples

```
library(vegan)
library(mgcv)
data(dune)
Ordination.model1 <- rda(dune)
plot1 <- ordiplot(Ordination.model1, choices=c(1,2), scaling=1)
ordicoeno(dune, ordiplot=plot1, legend=TRUE)
```

ordisymbol

Add Other Graphical Items to Ordination Diagrams

Description

Functions to add some other graphical itmes to ordination diagrams than provided within **vegan** by [ordihull](#), [ordispider](#), [ordiarrows](#), [ordisegments](#), [ordigrd](#), [ordiellipse](#), [ordiclust](#) and [lines.spantree](#).

Usage

```
ordisymbol(ordiplot, y, factor, col = 1, rainbow = TRUE,
  legend = TRUE, legend.x = "topleft", legend.ncol = 1, ...)
ordibubble(ordiplot,var,...)
ordiclust2(ordiplot, cluster, mingroups = 1, maxgroups = nrow(ordiplot$sites), ...)
ordinearest(ordiplot, dist,...)
ordivector(ordiplot, spec, lty=2,...)
```

Arguments

<code>ordiplot</code>	An ordination graph created by <code>ordiplot</code> (vegan).
<code>y</code>	Environmental data frame.
<code>factor</code>	Variable of the environmental data frame that defines subsets to be given different symbols.
<code>var</code>	Continuous variable of the environmental dataset or species from the community dataset.
<code>col</code>	Colour (as <code>points</code>).
<code>rainbow</code>	Use rainbow colours.
<code>legend</code>	Add the legend.
<code>legend.x</code>	Location of the legend; see also <code>legend</code> .
<code>legend.ncol</code>	the number of columns in which to set the legend items; see also <code>legend</code>
<code>cluster</code>	Cluster object.
<code>mingroups</code>	Minimum of clusters to be plotted.
<code>maxgroups</code>	Maximum of clusters to be plotted..
<code>dist</code>	Distance matrix.
<code>spec</code>	Species name from the community dataset.
<code>lty</code>	Line type as specified for <code>par</code> .
<code>...</code>	Other arguments passed to functions <code>points</code> , <code>symbols</code> , <code>ordihull</code> or <code>arrows</code> .

Details

Function `ordisymbol` plots different levels of the specified variable in different symbols and different colours (if `rainbow` option was selected).

Function `ordibubble` draws bubble diagrams indicating the value of the specified continuous variable. Circles indicate positive values, squares indicate negative values.

Function `ordiclust2` provides an alternative method of overlaying information from hierarchical clustering on an ordination diagram than provided by function `ordiclust`. The method draws convex hulls around sites that are grouped into the same cluster. You can select the minimum and maximum number of clusters that are plotted (i.e. the range of clustering steps to be shown).

Function `ordinearest` draws a vector from each site to the site that is nearest to it as determined from a distance matrix. When you combine the method with `lines.spantree` using the same distance measure, then you can evaluate in part how the minimum spanning tree was constructed.

Function `ordivector` draws a vector for the specified species on the ordination diagram and draws perpendicular lines from each site to a line that connects the origin and the head of species vector. This method helps in the biplot interpretation of a species vector as described by Jongman, ter Braak and van Tongeren (1995).

Value

These functions add graphical items to an existing ordination diagram.

Author(s)

Roeland Kindt (World Agroforestry Centre) and Jari Oksanen (ordinearest)

References

Jongman, R.H.G, ter Braak, C.J.F & van Tongeren, O.F.R. (1987). Data Analysis in Community and Landscape Ecology. Pudog, Wageningen.

Kindt, R. & Coe, R. (2005). Tree diversity analysis: A manual and software for common statistical methods for ecological and biodiversity studies.

<http://www.worldagroforestry.org/resources/databases/tree-diversity-analysis>

Examples

```
library(vegan)
data(dune)
data(dune.env)
Ordination.model1 <- rda(dune)
plot1 <- ordiplot(Ordination.model1, choices=c(1,2), scaling=2)
ordisymbol(plot1, dune.env, "Management", legend=TRUE,
            legend.x="topleft", legend.ncol=1)
plot2 <- ordiplot(Ordination.model1, choices=c(1,2), scaling=1)
distmatrix <- vegdist(dune, method='bray')
cluster <- hclust(distmatrix, method='single')
ordicluster2(plot2, cluster)
ordinearest(plot2, distmatrix, col=2)
ordivector(plot2, "Agrostol", lty=2)
```

PCAsignificance

PCA Significance

Description

Calculates the number of significant axes from a Principal Components Analysis based on the broken-stick criterion, or adds an equilibrium circle to an ordination diagram.

Usage

```
PCAsignificance(pca, axes=8)
ordiequilibriumcircle(pca, ordiplot, ...)
```

Arguments

pca	Principal Components Analysis result as calculated by <code>rda</code> (vegan).
axes	Number of axes to calculate results for.
ordiplot	Ordination plot created by <code>ordiplot</code> (vegan)
...	Other arguments passed to function <code>arrows</code> .

Details

These functions provide two methods of providing some information on significance for a Principal Components Analysis (PCA).

Function `PCAsignificance` uses the broken-stick distribution to evaluate how many PCA axes are significant. This criterion is one of the most reliable to check how many axes are significant. PCA axes with larger percentages of (accumulated) variance than the broken-stick variances are significant (Legendre and Legendre, 1998).

Function `ordiequilibriumcircle` draws an equilibrium circle to a PCA ordination diagram. Only species vectors with heads outside of the equilibrium circle significantly contribute to the ordination diagram (Legendre and Legendre, 1998). Vectors are drawn for these species. The function considers the scaling methods used by `rda` for `scaling=1`. The method should only be used for `scaling=1` and PCA calculated by function `rda`.

Value

Function `PCAsignificance` returns a matrix with the variances that are explained by the PCA axes and by the broken-stick criterion.

Function `ordiequilibriumcircle` plots an equilibrium circle and returns a list with the radius and the scaling constant used by `rda`.

Author(s)

Roeland Kindt (World Agroforestry Centre)

References

Legendre, P. & Legendre, L. (1998). Numerical Ecology. 2nd English Edition. Elsevier.

Kindt, R. & Coe, R. (2005). Tree diversity analysis: A manual and software for common statistical methods for ecological and biodiversity studies.

<http://www.worldagroforestry.org/resources/databases/tree-diversity-analysis>

Examples

```
library(vegan)
data(dune)
Ordination.model1 <- rda(dune)
PCAsignificance(Ordination.model1)
plot1 <- ordiplot(Ordination.model1, choices=c(1,2), scaling=1)
ordiequilibriumcircle(Ordination.model1,plot1)
```

radfitresult *Alternative Rank Abundance Fitting Results*

Description

Provides alternative methods of obtaining rank abundance curves than provided by functions `radfit`, `fisherfit` and `prestonfit` (**vegan**), although these same functions are called.

Usage

```
radfitresult(x,y="", factor="", level, plotit=T)
```

Arguments

x	Community data frame with sites as rows, species as columns and species abundance as cell values.
y	Environmental data frame.
factor	Variable of the environmental data frame that defines subsets to calculate fitted rank-abundance curves for.
level	Level of the variable to create the subset to calculate fitted rank-abundance curves.
plotit	Plot the results obtained by <code>plot.radfit</code> .

Details

These functions provide some alternative methods of obtaining fitted rank-abundance curves, although functions `radfit`, `fisherfit` and `prestonfit` (**vegan**) are called to calculate the actual results.

Value

The function returns the results from three methods of fitting rank-abundance curves:

<code>radfit</code>	results of <code>radfit</code> .
<code>fisherfit</code>	results of <code>fisherfit</code> .
<code>prestonfit</code>	results of <code>prestonfit</code> .

Optionally, a plot is provided of the `radfit` results by `plot.radfit`.

Author(s)

Roeland Kindt (World Agroforestry Centre)

References

Kindt, R. & Coe, R. (2005) Tree diversity analysis: A manual and software for common statistical methods for ecological and biodiversity studies.

<http://www.worldagroforestry.org/resources/databases/tree-diversity-analysis>

Examples

```
library(vegan)
data(BCI)
BCIall <- t(as.matrix(colSums(BCI)))
radfitresult(BCIall)
```

rankabundance	<i>Rank Abundance Curves</i>
---------------	------------------------------

Description

Provides methods of calculating rank-abundance curves.

Usage

```
rankabundance(x,y="", factor="", level, digits=1, t=qt(0.975, df=n-1))
rankabunplot(xr, addit=F, labels="", scale="abundance", scaledx=F, type="o",
             xlim=c(min(xpos), max(xpos)), ylim=c(0, max(x[, scale])), specnames=c(1:5), ...)
rankabuncomp(x,y="", factor, scale="abundance", scaledx=F, type="o", rainbow=T,
             legend=T, xlim=c(1, max1), ylim=c(0, max2), ...)
```

Arguments

x	Community data frame with sites as rows, species as columns and species abundance as cell values.
y	Environmental data frame.
factor	Variable of the environmental data frame that defines subsets to calculate rank abundance curves for.
level	Level of the variable to create the subset to calculate rank abundance curves.
digits	Number of digits in the results.
t	t-value to calculate confidence interval limits for the species proportion for cluster sampling (following Hayek and Buzas 1997).
xr	Result from rankabundance.
addit	Add rank abundance curve to an existing graph.
labels	Labels to plot at left of the rank abundance curves.
scale	Method of scaling the vertical axis. Method "abundance" uses abundance, "proportion" uses proportional abundance (species abundance / total abundance), "logabun" calculates the logarithm of abundance using base 10 and "accumfreq" accumulates the proportional abundance.
scaledx	Scale the horizontal axis to 100 percent of total number of species.
type	Type of plot (as in function plot)
xlim	Limits for the horizontal axis.
ylim	Limits for the vertical axis.

specnames	Vector positions of species names to add to the rank-abundance curve.
rainbow	Use rainbow colouring for the different curves.
legend	Add the legend (you need to click in the graph where the legend needs to be plotted).
...	Other arguments to be passed to functions <code>plot</code> or <code>points</code> .

Details

These functions provide methods of calculating and plotting rank-abundance curves.

The vertical axis can be scaled by various methods. Method "abundance" uses abundance, "proportion" uses proportional abundance (species abundance / total abundance), "logabun" calculates the logarithm of abundance using base 10 and "accumfreq" accumulates the proportional abundance.

The horizontal axis can be scaled by the total number of species, or by 100 percent of all species by option "scaledx".

The method of calculating the confidence interval for species proportion is described in Hayek and Buzas (1997).

Functions `rankabundance` and `rankabuncomp` allow to calculate rank abundance curves for subsets of the community and environmental data sets. Function `rankabundance` calculates the rank abundance curve for the specified level of a selected environmental variable. Method `rankabuncomp` calculates the rank abundance curve for all levels of a selected environmental variable separately.

Value

The functions provide information on rankabundance curves. Function `rankabundance` provides information on abundance, proportional abundance, logarithmic abundance and accumulated proportional abundance. The function also provides confidence interval limits for the proportion of each species (pflower, pupper) and the proportion of species ranks (in percentage).

Author(s)

Roeland Kindt (World Agroforestry Centre)

References

Hayek, L.-A. C. & Buzas, M.A. (1997). *Surveying Natural Populations*. Columbia University Press.

Kindt, R. & Coe, R. (2005) *Tree diversity analysis: A manual and software for common statistical methods for ecological and biodiversity studies*.

<http://www.worldagroforestry.org/resources/databases/tree-diversity-analysis>

Examples

```
library(vegan)
data(dune.env)
data(dune)
RankAbun.1 <- rankabundance(dune)
RankAbun.1
```

```

rankabunplot(RankAbun.1,scale='abundance', addit=FALSE, specnames=c(1,2,3))
rankabuncomp(dune, y=dune.env, factor='Management',
  scale='proportion', legend=FALSE)
## CLICK IN THE GRAPH TO INDICATE WHERE THE LEGEND NEEDS TO BE PLACED
## IF YOU OPT FOR LEGEND=TRUE.

```

removeNAcomm

Synchronize Community and Environmental Datasets

Description

These functions may assist to ensure that the sites of the community dataset are the same sites as those from the environmental dataset, something that is assumed to be the case for the **BiodiversityR** and **vegan** packages.

Usage

```

same.sites(x, y)
check.datasets(x, y)
check.ordiscores(x, ord, check.species = TRUE)
removeNAcomm(x, y, variable)
removeNAenv(x, variable)
removezerospecies(x)
subsetcomm(x, y, factor, level, returncomm = TRUE)

```

Arguments

x	Data frame assumed to be the community dataset with variables corresponding to species.
y	Data frame assumed to be the environmental dataset with variables corresponding to descriptors of sites.
ord	Ordination result.
check.species	Should the species scores be checked (TRUE) or not.
variable	Name of the variable from the environmental dataset with NA values that indicate those sites that should be removed.
factor	Variable of the environmental data frame that defines subsets for the data frame.
level	Level of the variable to create the subsets for the data frame.
returncomm	For the selected sites, return the community dataset (TRUE) or the environmental dataset.

Details

Function `same.sites` provides a new data frame that has the same row names as the row names of the environmental data set and the same (species) variables as the original community data set. Sites from the original community data set that have no corresponding sites in the environmental data set are not included in the new community data set. (Hint: this function can be especially useful when some sites do not contain any species and where a community dataset was generated by the `makecommunitydataset` function.)

Function `check.datasets` checks whether the community and environmental data sets have the same number of rows, and (if this was the case) whether the rownames of both data sets are the same. The function also returns the dimensions of both data sets.

Function `check.ordiscores` checks whether the community data set and the ordination result have the same number of rows (sites) and columns (species, optional for `check.species==TRUE`), and (if this was the case) whether the row and column names of both data sets are the same. Site and species scores for the ordination result are obtained via function `scores (vegan)`.

Functions `removeNAcomm` and `removeNAenv` provide a new data frame that does not contain NA for the specified variable. The specified variable is part of the environmental data set. These functions are particularly useful when using community and environmental datasets, as new community and environmental datasets can be calculated that contain information from the same sample plots (sites). An additional result of `removeNAenv` is that factor levels of any categorical variable that do not occur any longer in the new data set are removed from the levels of the categorical variable.

Function `removezerospecies` removes species from a community dataset that have total abundance that is smaller or equal to zero.

Function `subsetcomm` makes a subset of sites that contain a specified level of a categorical variable from the environmental data set. The same functionality of selecting subsets of the community or environmental data sets are implemented in various functions of **BiodiversityR** (for example `diversityresult`, `renyiresult` and `accumresult`) and have the advantage that it is not necessary to create a new data set. If a community dataset is returned, species that did not contain any individuals were removed from the data set. If an environmental dataset is returned, factor levels that did not occur were removed from the data set.

Value

The functions return a data frame or results of tests on the correspondence between community and environmental data sets.

Author(s)

Roeland Kindt (World Agroforestry Centre)

References

Kindt, R. & Coe, R. (2005) Tree diversity analysis: A manual and software for common statistical methods for ecological and biodiversity studies.

<http://www.worldagroforestry.org/resources/databases/tree-diversity-analysis>

Examples

```

library(vegan)
data(dune.env)
data(dune)
dune.env2 <- dune.env
dune.env2[1:4,"Moisture"] <- NA
dune2 <- removeNAcomm(dune,dune.env2,"Moisture")
dune.env2 <- removeNAenv(dune.env2,"Moisture")
dune3 <- same.sites(dune,dune.env2)
check.datasets(dune,dune.env2)
check.datasets(dune2,dune.env2)
check.datasets(dune3,dune.env2)
dune4 <- subsetcomm(dune,dune.env,"Management","NM",returncomm=TRUE)
dune.env4 <- subsetcomm(dune,dune.env,"Management","NM",returncomm=FALSE)
dune5 <- same.sites(dune,dune.env4)
check.datasets(dune4,dune5)

```

renyireult

*Alternative Renyi Diversity Results***Description**

Provides some alternative methods of obtaining results on Renyi diversity profile values than provided by [renyi](#) (**vegan**).

Usage

```
renyireult(x, y="", factor, level, method = "all",
          scales = c(0,0.25,0.5,1,2,4,8,Inf), evenness = F,...)
```

```
renyiplot(xr,addit=F, pch = 1,
          xlab = "alpha", ylab = "H-alpha", ylim = c(0,m),
          labelit = T, legend = T, col = 1, cex = 1,
          rainbow = T, evenness = F, ...)
```

```
renyiaccumresult(x, y = "", factor, level,
                 scales=c(0,0.25,0.5,1,2,4,8,Inf), permutations = 100,...)
```

```
renyicomp(x, y, factor, sites=Inf,
          scales = c(0,0.25,0.5,1,2,4,8,Inf), permutations = 100, plotit = T,...)
```

Arguments

x	Community data frame with sites as rows, species as columns and species abundance as cell values.
y	Environmental data frame.
factor	Variable of the environmental data frame that defines subsets to calculate diversity profiles for.

level	Level of the variable to create the subset to calculate diversity profiles.
method	Method of calculating the diversity profiles: "all" calculates the diversity of the entire community (all sites pooled together), "s" calculates the diversity of each site separately.
scales	Scale parameter values as in function renyi (vegan).
evenness	Calculate or plot the evenness profile.
xr	Result from renyi or <code>renyiresult</code> .
addit	Add diversity profile to an existing graph.
pch	Symbol used for drawing the diversity profiles (as in function points).
xlab	Label for the horizontal axis.
ylab	Label for the vertical axis.
ylim	Limits of the vertical axis.
labelit	Provide site labels (site names) at beginning and end of the diversity profiles.
legend	Add the legend (you need to click in the graph where the legend needs to be plotted).
col	Colour for the diversity profile (as in function points).
cex	Character expansion factor (as in function points).
rainbow	Use rainbow colours for the diversity profiles.
sites	Maximum number of sites to provide profile values.
permutations	Number of permutations for the Monte-Carlo simulations for accumulated renyi diversity profiles (estimated by renyiaccum).
plotit	Plot the results (you need to click in the graph where the legend should be plotted).
...	Other arguments to be passed to functions renyi or plot .

Details

These functions provide some alternative methods of obtaining results with diversity profiles, although function [renyi](#) is always used to calculate the diversity profiles.

The method of calculating the diversity profiles: "all" calculates the diversity profile of the entire community (all sites pooled together), whereas "s" calculates the diversity profile of each site separately. The evenness profile is calculated by subtracting the profile value at scale 0 from all the profile values.

Functions `renyiresult`, `renyiaccumresult` and `renyicom` allow to calculate diversity profiles for subsets of the community and environmental data sets. functions `renyiresult` and `renyiaccumresult` calculate the diversity profiles for the specified level of a selected environmental variable. Method `renyicom` calculates the diversity profile for all levels of a selected environmental variable separately.

Functions `renyicom` and `renyiaccumresult` calculate accumulation curves for the Renyi diversity profile by randomised pooling of sites and calculating diversity profiles for the pooled sites as implemented in [renyiaccum](#). The method is similar to the random method of species accumulation ([specaccum](#)). If the number of "sites" is not changed from the default, it is replaced by the sample size of the level with the fewest number of sites.

Value

The functions provide alternative methods of obtaining Renyi diversity profiles.

Author(s)

Roeland Kindt (World Agroforestry Centre)

References

Kindt R., Degrande A., Turyomurugyendo L., Mbosso C., Van Damme P., Simons A.J. (2001). Comparing species richness and evenness contributions to on-farm tree diversity for data sets with varying sample sizes from Kenya, Uganda, Cameroon and Nigeria with randomised diversity profiles. Paper presented at IUFRO conference on forest biometry, modeling and information science, 26-29 June, University of Greenwich, UK

Kindt R. (2002). Methodology for tree species diversification planning for African agroecosystems. Thesis submitted in fulfilment of the requirement of the degree of doctor (PhD) in applied biological sciences. Faculty of agricultural and applied biological sciences, Ghent University, Ghent (Belgium), 332+xi pp.

Kindt R., Van Damme P. & Simons A.J. (2006). Tree diversity in western Kenya: using diversity profiles to characterise richness and evenness. *Biodiversity and Conservation* 15: 1253-1270.

Kindt, R. & Coe, R. (2005) Tree diversity analysis: A manual and software for common statistical methods for ecological and biodiversity studies.

<http://www.worldagroforestry.org/resources/databases/tree-diversity-analysis>

Examples

```
library(vegan)
data(dune.env)
data(dune)
Renyi.1 <- renyiresult(dune, y=dune.env, factor='Management', level='NM',
  method='s')
Renyi.1
renyiplot(Renyi.1, evenness=FALSE, addit=FALSE, pch=1,col='1', cex=1,
  legend=FALSE)
## CLICK IN THE GRAPH TO INDICATE WHERE THE LEGEND NEEDS TO BE PLACED
## IN CASE THAT YOU OPT FOR LEGEND=TRUE
```

residualssurface

Show and Interpolate Two Dimensional Distribution of Residuals

Description

This function interpolates the spatial structure of residuals of a GLM through `gam` or `surf.ls` and optionally provides a graph.

Usage

```
residualssurface(model, data, x, y, gam = F, npol = 2, plotit = T, filled = F, bubble = F)
```

Arguments

model	Result of GLM as calculated by <code>glm</code> or <code>glm.nb</code> .
data	Data set that contains the spatial coordinates of the sample units used for the original model (specified as "x" and "y").
x	Horizontal position of the sample units.
y	Vertical position of the sample units.
gam	Interpolate the spatial structure by <code>gam</code> (if "TRUE") or by <code>surf.ls</code> (if "FALSE").
npol	Degree of polynomial surface as passed to <code>surf.ls</code> .
plotit	Plot the interpolated surface (through <code>interp</code> and the residuals).
filled	Fill the contours by <code>filled.contour</code> .
bubble	Provide a bubble graph of the residuals: circles indicate positive residuals, whereas squares indicate negative residuals.

Details

The function reports the results of a GAM or least-squares trend surface analysis of the spatial distribution of residuals of a model (through `residuals`).

Optionally, a graph is produced that can contain the trend surface, filled contours and bubble graphs in addition to the spatial location of the sample units.

Value

The function reports the results of a GAM or least-squares trend surface analysis of the spatial distribution of residuals. Optionally, a graph is provided.

Author(s)

Roeland Kindt (World Agroforestry Centre)

References

Kindt, R. & Coe, R. (2005) Tree diversity analysis: A manual and software for common statistical methods for ecological and biodiversity studies.

<http://www.worldagroforestry.org/resources/databases/tree-diversity-analysis>

Examples

```
library(vegan)
library(mgcv)
library(akima)
data(faramea)
Count.model1 <- lm(Faramea occidentalis ~ Precipitation,
  data=faramea, na.action=na.exclude)
```

```
surface.1 <- residualssurface(Count.model1, na.omit(faramea),
  'UTM.EW', 'UTM.NS', gam=TRUE, plotit=TRUE, bubble=TRUE)
```

spatialsample

Spatial Sampling within a Polygon

Description

Spatial sampling within a polygon provides several methods of selecting rectangular sample plots within a polygon. Using a GIS package may be preferred for actual survey design.

Usage

```
spatialsample(x,method="random",n=5,xwidth=0.5,ywidth=0.5,xleft=0,
  ylower=0,xdist=0,ydist=0,plotit=T,plothull=F)
```

Arguments

x	2-column matrix with the coordinates of the vertices of the polygon. The first column contains the horizontal (x) position, the second column contains the vertical (y) position.
method	Method of sampling, any of "random", "grid" or "random grid".
n	Number of sample plots to be selected, or number of horizontal and vertical grid positions.
xwidth	Horizontal width of the sample plots.
ywidth	Vertical width of the sample plots.
xleft	Horizontal starting position of the grid.
ylower	Vertical starting position of the grid.
xdist	Horizontal distance between grid locations.
ydist	Vertical distance between grid locations.
plotit	Plot the sample plots on the current graph.
plothull	Plot a convex hull around the sample plots.

Details

Spatial sampling within a polygon provides several methods of selecting the position of sample plots.

Method "random" selects random positions of the sample plots using simple random sampling.

Method "grid" selects sample plots from a grid defined by "xleft", "ylower", "xdist" and "ydist". In case `xdist=0` or `ydist=0`, then the number of grid positions are defined by "n". In case "xleft" or "ylower" are below the minimum position of any vertex of the polygon, then a random starting position is selected for the grid.

Method "random grid" selects sample plots at random from the sampling grid using the same methods of defining the grid as for method "grid".

Value

The function returns a list of centres of rectangular sample plots.

Author(s)

Roeland Kindt (World Agroforestry Centre)

References

Kindt, R. & Coe, R. (2005) Tree diversity analysis: A manual and software for common statistical methods for ecological and biodiversity studies.

<http://www.worldagroforestry.org/resources/databases/tree-diversity-analysis>

Examples

```
library(splancs)
area <- array(c(10,10,15,35,40,35,5,35,35,30,30,10), dim=c(6,2))
landuse1 <- array(c(10,10,15,15,30,35,35,30), dim=c(4,2))
landuse2 <- array(c(10,10,15,15,35,30,10,30,30,35,30,15), dim=c(6,2))
landuse3 <- array(c(10,10,30,35,40,35,5,10,15,30,30,10), dim=c(6,2))
plot(area[,1], area[,2], type="n", xlab="horizontal position",
      ylab="vertical position", lwd=2, bty="l")
polygon(landuse1)
polygon(landuse2)
polygon(landuse3)
spatialsample(area, method="random", n=20, xwidth=1, ywidth=1, plotit=TRUE,
              plothull=FALSE)
spatialsample(area, method="grid", xwidth=1, ywidth=1, plotit=TRUE, xleft=12,
              ylower=7, xdist=4, ydist=4)
spatialsample(area, method="random grid", n=20, xwidth=1, ywidth=1,
              plotit=TRUE, xleft=12, ylower=7, xdist=4, ydist=4)
```

transfgradient

Gradient for Hypothetical Example of Turnover of Species Composition

Description

This dataset documents the site sequence of 19 sites on a gradient determined from unimodal species distributions. The dataset is accompanied by [transfspecies](#) that documents the species composition of the sites. This is a hypothetical example that allows to investigate how well ecological distance measures or ordination methods recover the expected best sequence of sites.

Usage

```
data(transfgradient)
```

Format

A data frame with 19 observations on the following variable.

gradient a numeric vector

Source

Legendre, P. & Gallagher, E.D. (2001) Ecologically meaningful transformations for ordination of species data. *Oecologia* 129: 271-280.

References

Figure 3a.

Examples

```
data(transfspecies)
data(transfgradient)
plot(transfspecies[,1]~transfgradient[,1],xlab="gradient",
      ylab="species abundance",type="n",ylim=c(0.5,8.5))
for (i in 1:9) {points(transfgradient[,1],transfspecies[,i],type="o",pch=i)}
```

transfspecies

Hypothetical Example of Turnover of Species Composition

Description

This dataset documents the species composition of 19 sites that follow a specific sequence of sites as determined from unimodal species distributions. The dataset is accompanied by [transfgradient](#) that documents the gradient in species turnover. This is a hypothetical example that allows to investigate how well ecological distance measures or ordination methods recover the expected best sequence of sites.

Usage

```
data(transfspecies)
```

Format

A data frame with 19 observations on the following 9 variables.

species1 a numeric vector

species2 a numeric vector

species3 a numeric vector

species4 a numeric vector

species5 a numeric vector

species6 a numeric vector

species7 a numeric vector
 species8 a numeric vector
 species9 a numeric vector

Details

The example in the Tree Diversity Analysis manual only looks at the ecological distance from the first site. Hence, only the first 10 sites that share some species with this site should be selected.

This dataset enables investigations of how well ecological distance measures and ordination diagrams reconstruct the gradient (sequence of sites). The gradient expresses how the sites would be arranged based on their species composition.

Source

Legendre, P. & Gallagher, E.D. (2001) Ecologically meaningful transformations for ordination of species data. *Oecologia* 129: 271-280.

References

Figure 3a.

Examples

```
data(transfspecies)
data(transfgradient)
plot(transfspecies[,1]~transfgradient[,1],xlab="gradient",
      ylab="species abundance",type="n",ylim=c(0.5,8.5))
for (i in 1:9) {points(transfgradient[,1],transfspecies[,i],type="o",pch=i)}
```

warcom

Warburgia ugandensis AFLP Scores

Description

This data set contains scores for 185 loci for 100 individuals of the *Warburgia ugandensis* tree species (a medicinal tree species native to Eastern Africa). Since the data set is a subset of a larger data set that originated from a study of several *Warburgia* species, some of the loci did not produce bands for *W. ugandensis* (i.e. some loci only contain zeroes). This data set is accompanied by `warenv` that describes population and regional structure of the 100 individuals.

Usage

```
data(warcom)
```

Format

A data frame with 100 observations on the following 185 variables.

locus001 a numeric vector
locus002 a numeric vector
locus003 a numeric vector
locus004 a numeric vector
locus005 a numeric vector
locus006 a numeric vector
locus007 a numeric vector
locus008 a numeric vector
locus009 a numeric vector
locus010 a numeric vector
locus011 a numeric vector
locus012 a numeric vector
locus013 a numeric vector
locus014 a numeric vector
locus015 a numeric vector
locus016 a numeric vector
locus017 a numeric vector
locus018 a numeric vector
locus019 a numeric vector
locus020 a numeric vector
locus021 a numeric vector
locus022 a numeric vector
locus023 a numeric vector
locus024 a numeric vector
locus025 a numeric vector
locus026 a numeric vector
locus027 a numeric vector
locus028 a numeric vector
locus029 a numeric vector
locus030 a numeric vector
locus031 a numeric vector
locus032 a numeric vector
locus033 a numeric vector
locus034 a numeric vector
locus035 a numeric vector

locus036 a numeric vector
locus037 a numeric vector
locus038 a numeric vector
locus039 a numeric vector
locus040 a numeric vector
locus041 a numeric vector
locus042 a numeric vector
locus043 a numeric vector
locus044 a numeric vector
locus045 a numeric vector
locus046 a numeric vector
locus047 a numeric vector
locus048 a numeric vector
locus049 a numeric vector
locus050 a numeric vector
locus051 a numeric vector
locus052 a numeric vector
locus053 a numeric vector
locus054 a numeric vector
locus055 a numeric vector
locus056 a numeric vector
locus057 a numeric vector
locus058 a numeric vector
locus059 a numeric vector
locus060 a numeric vector
locus061 a numeric vector
locus062 a numeric vector
locus063 a numeric vector
locus064 a numeric vector
locus065 a numeric vector
locus066 a numeric vector
locus067 a numeric vector
locus068 a numeric vector
locus069 a numeric vector
locus070 a numeric vector
locus071 a numeric vector
locus072 a numeric vector

locus073 a numeric vector
locus074 a numeric vector
locus075 a numeric vector
locus076 a numeric vector
locus077 a numeric vector
locus078 a numeric vector
locus079 a numeric vector
locus080 a numeric vector
locus081 a numeric vector
locus082 a numeric vector
locus083 a numeric vector
locus084 a numeric vector
locus085 a numeric vector
locus086 a numeric vector
locus087 a numeric vector
locus088 a numeric vector
locus089 a numeric vector
locus090 a numeric vector
locus091 a numeric vector
locus092 a numeric vector
locus093 a numeric vector
locus094 a numeric vector
locus095 a numeric vector
locus096 a numeric vector
locus097 a numeric vector
locus098 a numeric vector
locus099 a numeric vector
locus100 a numeric vector
locus101 a numeric vector
locus102 a numeric vector
locus103 a numeric vector
locus104 a numeric vector
locus105 a numeric vector
locus106 a numeric vector
locus107 a numeric vector
locus108 a numeric vector
locus109 a numeric vector

locus110 a numeric vector
locus111 a numeric vector
locus112 a numeric vector
locus113 a numeric vector
locus114 a numeric vector
locus115 a numeric vector
locus116 a numeric vector
locus117 a numeric vector
locus118 a numeric vector
locus119 a numeric vector
locus120 a numeric vector
locus121 a numeric vector
locus122 a numeric vector
locus123 a numeric vector
locus124 a numeric vector
locus125 a numeric vector
locus126 a numeric vector
locus127 a numeric vector
locus128 a numeric vector
locus129 a numeric vector
locus130 a numeric vector
locus131 a numeric vector
locus132 a numeric vector
locus133 a numeric vector
locus134 a numeric vector
locus135 a numeric vector
locus136 a numeric vector
locus137 a numeric vector
locus138 a numeric vector
locus139 a numeric vector
locus140 a numeric vector
locus141 a numeric vector
locus142 a numeric vector
locus143 a numeric vector
locus144 a numeric vector
locus145 a numeric vector
locus146 a numeric vector

locus147 a numeric vector
locus148 a numeric vector
locus149 a numeric vector
locus150 a numeric vector
locus151 a numeric vector
locus152 a numeric vector
locus153 a numeric vector
locus154 a numeric vector
locus155 a numeric vector
locus156 a numeric vector
locus157 a numeric vector
locus158 a numeric vector
locus159 a numeric vector
locus160 a numeric vector
locus161 a numeric vector
locus162 a numeric vector
locus163 a numeric vector
locus164 a numeric vector
locus165 a numeric vector
locus166 a numeric vector
locus167 a numeric vector
locus168 a numeric vector
locus169 a numeric vector
locus170 a numeric vector
locus171 a numeric vector
locus172 a numeric vector
locus173 a numeric vector
locus174 a numeric vector
locus175 a numeric vector
locus176 a numeric vector
locus177 a numeric vector
locus178 a numeric vector
locus179 a numeric vector
locus180 a numeric vector
locus181 a numeric vector
locus182 a numeric vector
locus183 a numeric vector
locus184 a numeric vector
locus185 a numeric vector

Source

Muchugi, A.N. (2007) Population genetics and taxonomy of important medicinal tree species of the genus *Warburgia*. PhD Thesis. Kenyatta University, Kenya.

Examples

```
data(warcom)
```

warenv

Warburgia ugandensis Population Structure

Description

This data set contains population and regional locations for 100 individuals of the *Warburgia ugandensis* tree species (a medicinal tree species native to Eastern Africa). This data set is associated with `warcom` that contains scores for 185 AFLP loci.

Usage

```
data(warenv)
```

Format

A data frame with 100 observations on the following 4 variables.

`population` a factor with levels Kibale Kitale Laikipia Lushoto Mara

`popshort` a factor with levels KKIT KLAI KMAR TLUS UKIB

`country` a factor with levels Kenya Tanzania Uganda

`rift.valley` a factor with levels east west

Source

Muchugi, A.N. (2007) Population genetics and taxonomy of important medicinal tree species of the genus *Warburgia*. PhD Thesis. Kenyatta University, Kenya.

Examples

```
data(warenv)
```

Index

*Topic **datasets**

BCI.env, 9
farama, 81
ifri, 82
transfgradient, 108
transfspecies, 109
warcom, 110
warenv, 116

*Topic **multivariate**

accumresult, 4
add.spec.scores, 6
balanced.specaccum, 7
BiodiversityRGUI, 10
CAPdiscrim, 33
caprescale, 35
crosstabanalysis, 37
deviancepercentage, 38
dist.eval, 39
dist.zeroes, 40
distdisplayed, 41
disttransform, 43
diversityresult, 44
importancevalue, 83
loaded.citations, 85
makecommunitydataset, 86
multiconstrained, 87
nested.anova.dbrda, 89
NMSrandom, 90
nnetrandom, 92
ordicoeno, 93
ordisymbol, 94
PCAsignificance, 96
radfitresult, 98
rankabundance, 99
removeNAcomm, 101
renyiresult, 103
residualssurface, 105
spatialsample, 107

*Topic **package**

BiodiversityR-package, 3
.packages, 85

accumcomp, 15
accumcomp(accumresult), 4
accumplot, 15
accumplot(accumresult), 4
accumresult, 4, 8, 15, 102
add.spec.scores, 6, 23, 34
addLayer, 51, 71
adonis, 89
agnes, 30, 31
anosim, 31–33
Anova, 19, 21
anova.cca, 26, 27, 87, 88
anova.gam, 19, 21
anova.glm, 19, 21, 38
anova.lm, 19
anova.negbin, 38
arrows, 95, 96
as.dist, 11, 23, 27, 30, 32
av.plots, 20, 21

balanced.specaccum, 7
BCI, 9, 82
BCI.env, 9, 82
bioclim, 50, 70
BiodiversityR (BiodiversityR-package), 3
BiodiversityR-package, 3
BiodiversityRGUI, 3, 10
bioenv, 39, 40
bioenv.numeric(dist.eval), 39
box.cox.powers, 15
boxplot, 14

CAPdiscrim, 26, 27, 33
caprescale, 35
capscale, 26, 27, 33, 35, 36, 87–89
cascadeKM, 30, 31
cca, 23, 26, 27, 87, 88

- check.datasets (removeNAcomm), 101
- check.ordiscores (removeNAcomm), 101
- chisq.test, 20, 37
- citation, 85
- clara, 30
- clip.clust, 30, 31
- cmdscale, 6, 7, 22, 23, 33–35
- cophenetic, 30, 31
- cor, 6, 39, 42, 68
- coverscale, 43, 44
- cr.plots, 19, 21
- crosstabanalysis, 37
- cutree, 30

- daisy, 32
- dataType, 49, 59
- decorana, 23
- decostand, 43
- deviancepercentage, 38
- diana, 30, 31
- dist.eval, 39
- dist.zeros, 40
- distconnected, 39, 40
- distdisplayed, 23, 25, 27, 29, 41
- disttransform, 14, 43
- diversity, 44, 45
- diversitycomp, 16
- diversitycomp (diversityresult), 44
- diversityresult, 16, 44, 102
- domain, 50, 70
- drop1, 19, 21

- earth, 50, 52, 70, 72
- effect, 19, 21
- ensemble.accepted.categories
 (ensemble.dummy.variables), 55
- ensemble.batch, 46, 61
- ensemble.drop1 (ensemble.test), 64
- ensemble.dummy.variables, 55
- ensemble.formulae, 51, 71, 79
- ensemble.formulae (ensemble.test), 64
- ensemble.habitat.change
 (ensemble.raster), 58
- ensemble.mean (ensemble.batch), 46
- ensemble.raster, 46, 53, 54, 56, 58, 68, 74,
 75, 80
- ensemble.simplified.categories
 (ensemble.dummy.variables), 55
- ensemble.strategy (ensemble.test), 64

- ensemble.test, 46, 49, 53, 54, 56, 59–61, 64,
 79
- ensemble.test.splits, 46, 48, 50, 53, 54, 61
- ensemble.weights (ensemble.test), 64
- envfit, 24, 27, 28
- estimateR, 45
- evaluate, 59, 68, 69, 73, 75
- evaluation.strip.data, 78
- evaluation.strip.plot, 61
- evaluation.strip.plot
 (evaluation.strip.data), 78
- extent, 48, 59, 68, 79
- extract, 51, 53, 56, 59, 60, 67, 68, 71

- factor, 32
- fanny, 30
- faramea, 81
- fda, 50, 52, 70, 72
- filled.contour, 106
- fisher.alpha, 44, 45
- fisherfit, 98
- freq, 56

- gam, 18, 20, 42, 50, 52, 70, 72, 93, 94, 105, 106
- gam.check, 19, 21
- gam.control, 51, 71
- gbm, 49, 51, 69, 71
- gbm.step, 49, 51, 69, 71, 73, 74
- geoDist, 70
- glht, 20, 22
- glm, 18, 20, 38, 50, 52, 70, 72, 106
- glm.control, 51, 71
- glm.nb, 18, 38, 106
- glmmPQL, 19

- hclust, 30, 31

- identify.ordiplot, 23, 27
- ifri, 82, 85
- importancevalue, 83
- influence.plot, 20, 22
- initMDS, 91
- interp, 106
- isoMDS, 6, 91

- kfold, 48, 68, 74
- kgs, 31
- kmeans, 30
- KML, 49, 59

- ks.test, 15
- ksvm, 50, 52, 70, 72

- lda, 33, 34
- legend, 93, 95
- levels, 80
- levene.test, 19, 21
- lines.spantree, 23, 24, 27, 28, 94, 95
- lm, 18
- loaded.citations, 85

- mahal, 50, 52, 60, 70, 72, 73
- makecommunitydataset, 13, 86, 102
- mantel, 30, 31, 33, 42
- maxent, 49, 51, 67–69, 71
- mean, 80
- metaMDS, 6, 23, 25, 90, 91
- ModelEvaluation, 61
- mrpp, 32, 33
- multiconstrained, 26, 27, 87

- na.omit, 38
- nested.anova.dbrda, 89
- nested.npmanova (nested.anova.dbrda), 89
- NMSrandom, 6, 23, 90
- nnet, 50–52, 70–74, 92
- nnetrandom, 20, 92

- ordiarrows, 24, 28, 94
- ordibubble, 24, 28
- ordibubble (ordisymbol), 94
- ordicluster, 23, 24, 27, 28, 94, 95
- ordicluster2, 23, 25, 27, 29
- ordicluster2 (ordisymbol), 94
- ordicoeno, 25, 29, 93
- ordiellipse, 24, 28, 94
- ordiequilibriumcircle, 25
- ordiequilibriumcircle
(PCAsignificance), 96
- ordigrd, 94
- ordihull, 24, 28, 94, 95
- ordinearest, 23, 25, 27, 29
- ordinearest (ordisymbol), 94
- ordiplot, 23–25, 27–29, 34, 35, 41, 42, 93,
95, 96
- ordisegments, 24, 28, 94
- ordispider, 24, 28, 94
- ordisurf, 24, 28
- ordisymbol, 24, 28, 94

- ordivector, 24, 25, 28, 29
- ordivector (ordisymbol), 94

- pairs, 14
- pam, 30
- par, 93, 95
- PCAsignificance, 96
- permutest.cca, 27
- persp.renyiaccum, 18
- plot, 5, 14, 21, 23, 27, 79, 93, 99, 100, 104
- plot.agnes, 31
- plot.cascadeKM, 31
- plot.cca, 23, 25, 27, 29
- plot.diana, 31
- plot.gam, 19, 21
- plot.hclust, 31
- plot.lm, 19, 21
- plot.prc, 27
- plot.radfit, 98
- plot.rpart, 20, 22
- plot.specaccum, 4, 5, 7
- points, 5, 16, 93, 95, 100, 104
- points.ordiplot, 24, 28
- postMDS, 6
- prc, 26, 27
- prcomp, 22
- predict, 19, 21, 48, 59, 79
- predict.rpart, 19
- prepareData, 51, 53, 56, 59, 60, 67, 68, 71, 79
- prestonfit, 98

- qq.plot, 15, 19, 21

- radfit, 98
- radfitresult, 17, 98
- randomForest, 50–52, 69, 71
- randomPoints, 48, 51, 59, 67, 68, 71, 79
- rankabuncomp, 17
- rankabuncomp (rankabundance), 99
- rankabundance, 17, 99
- rankabunplot, 17
- rankabunplot (rankabundance), 99
- rankindex, 32
- raster, 55, 56
- rda, 6, 7, 22, 23, 25–27, 29, 87, 88, 96, 97
- rect.hclust, 30, 31
- removeNAcomm, 13, 101
- removeNAenv, 13
- removeNAenv (removeNAcomm), 101

- removezerospecies (removeNAcomm), 101
- renyi, 103, 104
- renyiaccum, 104
- renyiaccumresult, 17, 18
- renyiaccumresult (renyiresult), 103
- renyicomp, 17, 18
- renyicomp (renyiresult), 103
- renyiplot, 18
- renyiplot (renyiresult), 103
- renyiresult, 17, 18, 102, 103
- residuals, 106
- residuals.rpart, 19
- residualssurface, 105
- round, 73
- rpart, 19, 20, 50, 52, 70, 72
- rpart.control, 52, 72

- s, 52, 72
- same.sites, 13
- same.sites (removeNAcomm), 101
- sammon, 6, 91
- scale, 19, 20
- scores, 23, 25, 27, 29, 102
- screplot.cca, 25
- shapiro.test, 15
- sink, 48, 59, 68
- spatialsample, 107
- specaccum, 4, 5, 7, 8, 104
- specnumber, 44, 45
- specpool, 44, 45
- stack, 48, 59, 67, 79
- step.gam, 50, 52, 70, 72
- stepAIC, 50, 52, 70, 72
- stressplot, 25
- subsetcomm (removeNAcomm), 101
- summary, 14
- summary.agnes, 30
- summary.anosim, 31
- summary.bioenv.numeric (dist.eval), 39
- summary.cca, 23, 27
- summary.clara, 30
- summary.decorana, 23
- summary.diana, 30
- summary.fanny, 30
- summary.gam, 19, 20
- summary.glm, 19, 20
- summary.lm, 19
- summary.nnet, 20
- summary.pam, 30

- summary.prc, 27
- summary.rpart, 20
- surf.ls, 105, 106
- svm, 50, 52, 70, 72
- symbols, 95

- termplot, 19, 21
- text.ordiplot, 24, 27
- text.rpart, 20, 22
- threshold, 49, 68, 69
- transfgradient, 108, 109
- transfspecies, 108, 109
- trunc, 60

- vegdist, 22, 23, 30, 32, 33, 39–42, 87, 89, 91
- vif, 68

- warcom, 110
- warenv, 116
- wascores, 6
- writeFormats, 48, 59, 72
- writeRaster, 48, 49, 56, 59, 72