Package ‘BiodiversityR’

April 20, 2021

Type Package
Title Package for Community Ecology and Suitability Analysis
Version 2.13-1
Date 2021-4-15
Author Roeland Kindt [cre, aut] (<https://orcid.org/0000-0002-7672-0712>)
Maintainer Roeland Kindt <R.KINDT@CGIAR.ORG>
Description Graphical User Interface (via the R-Commander) and 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-3
URL http://www.worldagroforestry.org/output/tree-diversity-analysis
Depends R (>= 3.2.2), tcltk, vegan (>= 2.5-1)
Imports Rcmdr (>= 2.6-1)
Suggests vegan3d, rgl, permute, lattice, MASS, mgcv, cluster, car, RODBC, rpart, effects, multcomp, ellipse, maptree, sp, splines, spatial, akima, nnet, dismo, raster (>= 2.8-19), rgdal, maxlike, gbm, randomForest, gam (>= 1.15), earth, mda, kernlab, e1071, glmnet, tools, methods, bootstrap, PresenceAbsence, geosphere, maptools, ENMeval, red, rgeos, igr aph, Rlof, maxnet, party, readxl, colorspace, dplyr, rlang, sf, blockCV
NeedsCompilation no
Repository CRAN
Date/Publication 2021-04-20 14:20:13 UTC

R topics documented:

BiodiversityR-package .................................................. 3
accumresult .......................................................... 4
add.spec.scores ......................................................... 7
balanced.specaccum .................................................. 9
BCI.env ................................................................. 10
BiodiversityR.changeLog ............................................. 11
BiodiversityRGUI ....................................................... 12
CAPdiscrim .............................................................. 13
caprescale ............................................................ 16
crosstabanalysis ........................................................ 17
deviancepercentage .................................................... 18
dist.eval ................................................................. 20
dist.zeros ............................................................... 21
distdisplayed ........................................................... 22
disttransform ........................................................... 24
diversityresult .......................................................... 25
ensemble.analogue ..................................................... 28
ensemble.batch ........................................................ 32
ensemble.bioclim ....................................................... 45
ensemble.bioclim.graph ................................................. 49
ensemble.calibrate.models .......................................... 52
ensemble.dummy.variables .......................................... 70
ensemble.ecocrop ..................................................... 73
ensemble.evaluate ..................................................... 76
ensemble.novel ........................................................ 79
ensemble.PET.season .................................................. 83
ensemble.PET.seasons ................................................ 85
ensemble.raster ........................................................ 88
ensemble.red .......................................................... 95
ensemble.spatialBlock ............................................... 100
ensemble.spatialThin .................................................. 104
ensemble.zones ........................................................ 109
evaluation.strip.data .................................................. 113
faramea ................................................................. 116
ifri ....................................................................... 117
importancevalue ....................................................... 118
loaded.citations ....................................................... 121
makecommunitydataset ............................................... 121
multiconstrained ....................................................... 123
nested.anova.dbRda .................................................... 125
NMSrandom ............................................................ 126
nnetrandom ............................................................ 128
ordicoeno ............................................................... 129
ordisymbol ............................................................. 130
PCAsignificance ....................................................... 133
radfitresult ............................................................. 134
rankabundance ......................................................... 135
removeNAcomm ......................................................... 138
renyiresult ............................................................. 141
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

https://www.worldagroforestry.org/output/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', cex.lab=1, cex.axis=1, ...)

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', cex.lab=1, cex.axis=1, ...)

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 `specaccum`). 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 `specaccum`).

Conditioned

Estimation of standard deviation is conditional on the empirical dataset for the exact SAC (as in function `specaccum`).

Gamma

Method for estimating the total extrapolated number of species in the survey area (as in `specaccum`).

Addit

Add species accumulation curve to an existing graph.

Xr

Result from `specaccum` or `accumresult`.

Col

Colour for drawing lines of the species accumulation curve (as in function `plot.specaccum`).

Labels

Labels to plot at left and right of the species accumulation curves.

Ci

Multiplier used to get confidence intervals from standard deviation (as in function `plot.specaccum`).

Pch

Symbol used for drawing the species accumulation curve (as in function `points`).

Type

Type of plot (as in function `plot`).

Cex

Character expansion factor (as in function `plot`).

Xlim

Limits for the X = horizontal axis.

Ylim

Limits for the Y = vertical axis.

Xlab

Label for the X = horizontal axis (as in function `title`).

Ylab

Label for the Y = vertical axis (as in function `title`).

Cex.lab

The magnification to be used for X and Y labels relative to the current setting of cex. (as in function `par`).

Cex.axis

The magnification to be used for axis annotation relative to the current setting of cex (as in function `par`).

Plotit

Plot the results.

Labelit

Label the species accumulation curves with the levels of the categorical variable.

Legend

Add the legend (you need to click in the graph where the legend needs to be plotted).

Rainbow

Use rainbow colouring for the different curves.

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

https://www.worldagroforestry.org/output/tree-diversity-analysis
https://rpubs.com/Roeland-KINDT

See Also

accumcomp.long

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

# Click in the graph to indicate where the legend needs to be placed for
# option where legend=TRUE (default).

# Not run:
# ggplot2 plotting method
data(warcom)
data(warenv)
Accum.3 <- accumcomp(warcom, y=warenv, factor='population',
add.spec.scores

method='exact', conditioned=F, plotit=F)

library(ggplot2)

# possibly need for extrafont::loadfonts(device="win") to have Arial
# as alternative, use library(ggThemeAssist)
BioR.theme <- theme(
  panel.background = element_blank(),
  panel.border = element_blank(),
  panel.grid = element_blank(),
  axis.line = element_line("gray25"),
  text = element_text(size = 12, family="Arial"),
  axis.text = element_text(size = 10, colour = "gray25"),
  axis.title = element_text(size = 14, colour = "gray25"),
  legend.title = element_text(size = 14),
  legend.text = element_text(size = 14),
  legend.key = element_blank())

accum.long3 <- accumcomp.long(Accum.3, ci=NA, label.freq=5)

plotgg1 <- ggplot(data=accum.long3, aes(x = Sites, y = Richness, ymax = UPR, ymin=LWR)) +
  scale_x_continuous(expand=c(0, 1), sec.axis = dup_axis(labels=NULL, name=NULL)) +
  scale_y_continuous(sec.axis = dup_axis(labels=NULL, name=NULL)) +
  geom_line(aes(colour=Grouping), size=2) +
  geom_point(data=subset(accum.long3, labelit==TRUE),
             aes(colour=Grouping, shape=Grouping), size=5) +
  geom_ribbon(aes(colour=Grouping), alpha=0.2, show.legend=FALSE) +
  BioR.theme +
  scale_color_brewer(palette = "Set1") +
  labs(x = "Trees", y = "Loci", colour = "Population", shape = "Population")

plotgg1

## End(Not run) # dontrun

---

**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")
```
add.spec.scores

Arguments

ordi Ordination result as calculated by cmdscale, isoMDS, sammon, postMDS, metaMDS or NMSrandom.

comm Community data frame with sites as rows, species as columns and species abundance as cell values.

method Method for calculating species scores. Method "cor.scores" calculates the scores by the correlation between site scores and species vectors (via function cor), method "wa.scores" calculates the weighted average scores (via function wascores) and method "pcoa.scores" calculates the scores by weighing the correlation between site scores and species vectors by variance explained by the ordination axes.

multi Multiplier for the species scores.

Rscale Use the same scaling method used by vegan for rda.

scaling Scaling method as used by rda.

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


https://www.worldagroforestry.org/output/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)
# Change scores for second axis
Ordination.model1$points[,2] <- -1.0 * Ordination.model1$points[,2]
Ordination.model1 <- add.spec.scores(Ordination.model1, dune, method='pcoa.scores', Rscale=TRUE, scaling=1, multi=1)
# Compare Ordination.model1 with PCA
Ordination.model2 <- rda(dune, scale=FALSE)
#
par(mfrow=c(1,2))
balanced.specaccum

ordiplot(Ordination.model1, type="text")
abline(h = 0, lty = 3)
abline(v = 0, lty = 3)
plot(Ordination.model2, type="text", 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

comm: Community data frame with sites as rows, species as columns and species abundance as cell values.

permutations: Number of permutations to calculate the species accumulation curve.

strata: Categorical variable used to specify strata.

grouped: Should sites from the same stratum be kept together (TRUE) or not.

reps: Number of subsamples to be taken from each stratum (see details).

scale: 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


Examples

```r
library(vegan)
data(dune.env)
data(dune)

# not balancing species accumulation
Accum.orig <- specaccum(dune)
Accum.orig

# randomly sample 3 quadrats from each stratum of Management
Accum.1 <- balanced.specaccum(dune, strata=dune.env$Management, reps=3)
Accum.1

# scale results by number of trees per quadrat
mean(dune.env$site.totals) <- apply(dune,1,sum)
Accum.2 <- balanced.specaccum(dune, strata=dune.env$Management, reps=3, scale=dune.env$site.totals)
Accum.2
```

---

**BCI.env**  
*Barro Colorado Island Quadrat Descriptions*

**Description**

Topography-derived variables and UTM coordinates 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

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

Format

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

- **UTM.EW**: UTM easting
- **UTM.NS**: UTM northing
- **elevation**: mean of the elevation values of the four cell corners
- **convex**: mean elevation of the target cell minus the mean elevation of the eight surrounding cells
- **slope**: mean angular deviation from horizontal of each of the four triangular planes formed by connecting three of its corners
- **aspectEW**: the sine of aspect
- **aspectNS**: the cosine of aspect

References


Examples

```r
data(BCI.env)
```
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). In 2012 methods for ensemble suitability 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 accommodates (including examples) is provided in the freely available Tree Diversity Analysis manual (Kindt and Coe, 2005) that is accessible via the help menu.

Usage
BiodiversityRGUI(changeLog = FALSE, backward.compatibility.messages = FALSE)

Arguments
changeLog Show the changeLog file
backward.compatibility.messages Some notes on backward compatibility

Details
The function launches the R-Commander GUI with an extra menu for common statistical methods for biodiversity and community ecology analysis as described in the Tree Diversity Analysis manual of Roeland Kindt and Richard Coe (available via https://www.worldagroforestry.org/output/tree-diversity-analysis) and expanded systematically with new functions that became available from the vegan community ecology package.

Since 2012, functions for ensemble suitability modelling were included in BiodiversityR. In 2016, a GUI was created for ensemble suitability modelling.

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 the Rcmdr, it is possible that newest 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. However, please alert Roeland Kindt about the issue.

BiodiversityR uses two data sets for biodiversity and community ecology 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 \texttt{as.dist} prior to further analysis).

For ensemble suitability modelling, different data sets should be created and declared such as the calibration stack, the presence data set and the absence data set. The ensemble suitability modelling menu gives some guidelines on getting started with ensemble suitability modelling.

**Value**

Besides launching the graphical user interface, the function gives some notes on backward compatibility.

**Author(s)**

Roeland Kindt (with some help from Jari Oksanen)

**References**


https://www.worldagroforestry.org/output/tree-diversity-analysis

---

**CAPdiscrim**

*Canonical Analysis of Principal Coordinates based on Discriminant Analysis*

**Description**

This function provides a method for CAP that follows the procedure as described by the authors of the ordination method (Anderson & Willis 2003). The CAP method implemented in \texttt{vegan} through \texttt{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, mmax=10, add=FALSE, 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 `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.

axes Number of PCoA axes (cmdscale) to provide in the result.

m Number of PCoA axes to be investigated by discriminant analysis (lda). 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).

mmax The maximum number of PCoA axes considered when searching (m=0) for the number of axes that provide the best classification success.

add Add a constant to the non-diagonal dissimilarities such that the modified dissimilarities are Euclidean; see also cmdscale.

permutations The number of permutations for significance testing.

**Details**

This function provides a method of Constrained Analysis of Principal Coordinates (CAP) that follows 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.

Results may be different than those obtained in the PRIMER-e package because PRIMER-e does not consider prior probabilities, does not standardize PCOA axes by their eigenvalues and applies an additional spherical standardization to a common within-group variance/covariance matrix.

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

- **percent.level**
  the percentage of correct predictions for different factor levels

- **x**
  the positions of the sites provided by the discriminant analysis

- **F**
  the squares of the singulare 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


https://www.worldagroforestry.org/output/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, add=FALSE)
Ordination.model1
plot1 <- ordiplot(Ordination.model1, type="none")
ordisymbol(plot1, dune.env, "Management", legend=TRUE)

# plot change in classification success against m
plot(seq(1:14), rep(-1000, 14), xlim=c(1, 14), ylim=c(0, 100), xlab="m",
ylab="classification success (percent)", type="n")
for (mseq in 1:14) {
  CAPdiscrim.result <- CAPdiscrim(dune~Management, data=dune.env,
  dist="bray", axes=2, m=mseq)
  points(mseq, CAPdiscrim.result$percent)
}

#
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


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$points[,1:14],method="euc"))
totalsumsquares1 <- sum(Distmatrix.2^2)/(2*20)
# Sum of distances among sites in principal coordinates analysis on axes
# corresponding to positive eigenvalues
totalsumsquares1
# [1] 4.593946896588808
Ordination.model2 <- capscale(dune ~ Management,dune.env,dist="bray", add=FALSE)
# Total sums of positive eigenvalues of the distance-based redundancy analysis
Ordination.model2$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**

```r
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


https://www.worldagroforestry.org/output/tree-diversity-analysis

Examples

```r
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

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

\begin{itemize}
\item \textbf{x} \hspace{1cm} Result of GLM as calculated by \texttt{glm} or \texttt{glm.nb}.
\item \textbf{data} \hspace{1cm} Data set to be used for the null model (preferably the same data set used by the 'full' model).
\item \textbf{test} \hspace{1cm} Test statistic to be used for the comparison between the null model and the 'full' model as estimated by \texttt{anova.glm} or \texttt{anova.negbin}: partial match of one of "Chisq", "F" or "Cp".
\item \textbf{digits} \hspace{1cm} Number of digits in the calculation of the percentage.
\end{itemize}

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 \texttt{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

https://www.worldagroforestry.org/output/tree-diversity-analysis

Examples

\begin{verbatim}
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)
\end{verbatim}
dist.eval

Distance Matrix Evaluation

Description

Function dist.eval provides one test of a distance matrix, and then continues with distconnected (vegan). Function prepare.bioenv converts selected variables to numeric variables and then excludes all categorical variables in preparation of applying bioenv (vegan).

Usage

dist.eval(x, dist)
prepare.bioenv(env, as.numeric = c())

Arguments

x Community data frame with sites as rows, species as columns and species abundance as cell values.
env Environmental data frame with sites as rows and variables as columns.
dist Method for calculating ecological distance with function vegdist: partial match to "manhattan", "euclidean", "canberra", "clark", "bray", "kulczynski", "jaccard", "gower", "morisita", "horn" or "mountford".

as.numeric 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 prepare.bioenv 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)
**dist.zeroses**

**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**

```r
dist.zeroses(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 `vegdist`.

**Examples**

```r
library(vegan)
data(dune)
dist.eval(dune, "euclidean")
dist.eval(dune, "bray")
## Not run:
data(dune.env)
dune.env2 <- dune.env[, c(\'A1\', \'Moisture\', \'Manure\')]
dune.env2$Moisture <- as.numeric(dune.env2$Moisture)
dune.env2$Manure <- as.numeric(dune.env2$Manure)
sol <- bioenv(dune ~ A1 + Moisture + Manure, dune.env2)
sol
summary(sol)
dune.env3 <- prepare.bioenv(dune.env, as.numeric=c(\'Moisture\', \'Manure\'))
bioenv(dune, dune.env3)
## End(Not run)
```
Details
This function 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
https://www.worldagroforestry.org/output/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")
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
```r
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 calculated 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


https://www.worldagroforestry.org/output/tree-diversity-analysis
Examples

```r
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)
```

---

**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`. For method "dispweight" a call is made to function `dispweight`.

**Usage**

```r
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" and "chi.square", "log", "square", "pa", "Braun.Blanquet", "Domin", "Hult", "Hill", "fix" and "coverscale.log" and "dispweight".

**Details**

This function 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.


Method "dispweight" uses function `dispweight` without specifying a grouping structure.
**diversityresult**

**Value**

The function returns a transformed community matrix.

**Author(s)**

Roeland Kindt (World Agroforestry Centre)

**References**


https://www.worldagroforestry.org/output/tree-diversity-analysis

**Examples**

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

**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**

```r
diversityvariables(x, y, digits=8)
diversitycomp(x, y = NULL,
```
factor1 = NULL, factor2 = NULL, 
"richness", "abundance", "Jevenness", "Eevenness", 
"jack1", "jack2", "chao", "boot"), 
method=c("pooled", "mean", "sd", "max", "jackknife"), 
sortit=FALSE, digits=8)

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, "Eevenness" 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: "pooled" calculates the diversity of the entire community (all sites pooled), "each site" calculates diversity for each site separately, "mean" calculates the average diversity of the sites, "sd" calculates the standard deviation of the diversity of the sites, "max" calculates the maximum diversity of the sites, whereas "jackknife" calculates the jackknifed diversity for the entire data frame.
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.

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.
Function `diversityvariables` adds variables to the environmental dataset (richness, Shannon, Simpson, inverseSimpson, Logalpha, Berger, Jevenness, Eevenness).

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

https://www.worldagroforestry.org/output/tree-diversity-analysis

Examples

```r
library(vegan)
data(dune.env)
data(dune)

diversityresult(dune, y=NULL, index="Shannon", method="each site", sortit=TRUE, digits=5)
diversityresult(dune, y=dune.env, factor="Management", level="NM", index="Shannon", method="each site", sortit=TRUE, digits=5)
diversityresult(dune, y=NULL, index="Shannon", method="pooled", digits=5)
```
diversityresult(dune, y=dune.env, factor="Management", level="NM", index="Shannon", method="pooled", digits=5)
diversityresult(dune, y=NULL, index="Shannon", method="mean", digits=5)
diversityresult(dune, y=NULL, index="Shannon", method="sd", digits=5)
diversityresult(dune, y=NULL, index="Shannon", method="jackknife", digits=5)
diversityresult(dune, y=dune.env, factor="Management", level="NM", index="Shannon", method="jackknife", digits=5)
diversitycomp(dune, y=dune.env, factor1="Moisture", index="Shannon", method="pooled", sortit=TRUE)
diversitycomp(dune, y=dune.env, factor1="Moisture", index="Shannon", method="mean", sortit=TRUE)
diversitycomp(dune, y=dune.env, factor1="Management", index="Shannon", method="jackknife", sortit=TRUE)
diversitycomp(dune, y=dune.env, factor1="Management", factor2="Moisture", index="Shannon", method="pooled", digits=6)
diversitycomp(dune, y=dune.env, factor1="Management", factor2="Moisture", index="Shannon", method="mean", digits=6)

ensemble.analogue
Climate analogues from climatic distance raster layers.

Description
Function ensemble.analogue creates the map with climatic distance and provides the locations of the climate analogues (defined as locations with smallest climatic distance to a reference climate). Function ensemble.analogue.object provides the reference values used by the prediction function used by predict.

Usage
ensemble.analogue(x = NULL, analogue.object = NULL, analogues = 1, RASTER.object.name = analogue.object$name, RASTER.stack.name = x@title, RASTER.format = "raster", RASTER.datatype = "INT2S", RASTER.NAflag = -32767, KML.out = T, KML.blur = 10, KML.maxpixels = 100000, limits = c(1, 5, 20, 50), limit.colours = c('red', 'orange', 'blue', 'grey'), CATCH.OFF = FALSE)

ensemble.analogue.object(ref.location, future.stack, current.stack, name = "reference1", method = "mahal", an = 10000, probs = c(0.025, 0.975), weights = NULL, z = 2)
Arguments

x: RasterStack object (stack) containing all environmental layers (climatic variables) for which climatic distance should be calculated.

analogue.object: Object listing reference values for the environmental layers and additional parameters (covariance matrix for method = "mahal" or normalization parameters for method = "quantile") that are used by the prediction function that is used internally by predict. This object is created with ensemble.analogue.object.

analogues: Number of analogue locations to be provided

RASTER.object.name: First part of the names of the raster file that will be generated, expected to identify the area and time period for which ranges were calculated

RASTER.stack.name: Last part of the names of the raster file that will be generated, expected to identify the predictor stack used

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 TRUE, then kml files will be saved in a subfolder 'kml/zones'.

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.

limits: Limits indicating the accumulated number of closest analogue sites. These limits will correspond to different colours in the KML map. In the default setting, the closest analogue will be coloured red and the second to fifth closest analogues will be coloured orange.

limit.colours: Colours for the different limits based on number of analogues.

CATCH.OFF: Disable calls to function tryCatch.

ref.location: Location of the reference location for which analogues are searched for and from which climatic distance will be calculated, typically available in 2-column (lon, lat) dataframe; see also extract.

future.stack: RasterStack object (stack) containing the environmental layers (climatic variables) to obtain the conditions of the reference location. For climate change research, this RasterStack object corresponds to the future climatic conditions of the reference location.

current.stack: RasterStack object (stack) containing all environmental layers (climatic variables) for which climatic distance should be calculated. For climate change research, this RasterStack object corresponds to the current climatic conditions and range where climate analogues are searched for.

name: Name of the object, expect to expected to identify the area and time period for which ranges were calculated and where no novel conditions will be detected.
Method used to calculate climatic distance: method = "mahal" results in using the Mahalanobis distance (mahalanobis); method = "quantile" results in dividing the differences between reference climatic values and climatic values in the 'current' raster by a quantile range obtained from the 'current' raster; method = "sd" results in dividing the differences between reference climatic values and climatic values in the 'current' raster by standard deviations obtained from the 'current' raster; and method = "none" results in not dividing these differences.

Number of randomly selected locations points to calculate the covariance matrix (cov) to be used with mahalanobis, therefore only used for method = "mahal". See also randomPoints.

Numeric vector of probabilities [0,1] as used by quantile). Only used for method = "quantile".

Numeric vector of weights by which each variable (difference) should be multiplied by (can be used to give equal weight to 12 monthly rainfall values and 24 minimum and maximum monthly temperature values). Not used for method = "mahal".

Parameter used as exponent for differences calculated between reference climatic variables and variables in the 'current' raster and reciprocal exponent for the sum of all differences. Default value of 2 corresponds to the Euclidean distance. Not used for method = "mahal".

Function ensemble.analogues maps the climatic distance from reference values determined by ensemble.analogues.object and provides the locations of the analogues closest analogues.

The method = "mahal" uses the Mahalanobis distance as environmental (climatic) distance: mahalanobis.

Other methods use a normalization method to handle scale differences between environmental (climatic) variables:

$$\text{ClimaticDistance} = \left(\sum_{i}(weight_i * (|T_i - C_i|/norm_i)^z)\right)^{1/z}$$

where $T_i$ are the target values for environmental (climatic) variable $i$, $C_i$ are the values in the current environmental layers where analogues are searched for, $weight_i$ are the weights for environmental variable $i$, and $norm_i$ are the normalization parameters for environmental variable $i$.

Function ensemble.analogue.object returns a list with following objects:

- **name**: name for the reference location
- **ref.location**: coordinates of the reference location
- **stack.name**: name for time period for which values are extracted from the future.stack
- **method**: method used for calculating climatic distance
- **target.values**: target environmental values to select analogues for through minimum climatic distance
- **cov.mahal**: covariance matrix
ensemble.analogue

norm.values  parameters by which each difference between target and 'current' value will be divided
weight.values  weights by which each difference between target and 'current' value will be multiplied
z  parameter to be used as exponent for differences between target and 'current' values

Author(s)

Roeland Kindt (World Agroforestry Centre) and Eike Luedeling (World Agroforestry Centre)

References


See Also

ensemble.novel

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)
predictors <- subset(predictors, subset=c("bio1", "bio5", "bio6", "bio7", "bio8",
  "bio12", "bio16", "bio17"))
predictors
predictors$title <- "base"

# instead of searching for current analogue of future climate conditions,
# search for analogue in southern hemisphere
future.stack <- stack(crop(predictors, y=extent(-125, -32, 0, 40)))
future.stack$title <- "north"
current.stack <- stack(crop(predictors, y=extent(-125, -32, -56, 0)))
current.stack$title <- "south"

# reference location in Florida
# in this case future.stack and current.stack are both current
ref.loc <- data.frame(t(c(-80.19, 25.76)))
names(ref.loc) <- c("lon", "lat")

# climate analogue analysis based on the Mahalanobis distance
Florida.object.mahal <- ensemble.analogue.object(ref.location=ref.loc,
  future.stack=future.stack, current.stack=current.stack,
Florida.analogue.mahal <- ensemble.analogue(x=current.stack,
analogue.object=Florida.object.mahal, analogues=50)

# climate analogue analysis based on the Euclidean distance and dividing each variable by the sd
Florida.object.sd <- ensemble.analogue.object(ref.location=ref.loc,
future.stack=future.stack, current.stack=current.stack,
name="FloridaSD", method="sd", z=2)

Florida.analogue.sd <- ensemble.analogue(x=current.stack,
analogue.object=Florida.object.sd, analogues=50)

# plot analogues on climatic distance maps
par(mfrow=c(1,2))
analogue.file <- paste(getwd(), '//ensembles//analogue//FloridaMahal_south_analogue.grd', sep="")
plot(raster(analogue.file), main="Mahalanobis climatic distance")
points(Florida.analogue.mahal[3:50, "lat"] ~ Florida.analogue.mahal[3:50, "lon"],
pch=1, col="red", cex=1)
points(Florida.analogue.mahal[3:50, "lat"] ~ Florida.analogue.mahal[3:50, "lon"],
pch=3, col="black", cex=1)
points(Florida.analogue.mahal[2, "lat"] ~ Florida.analogue.mahal[2, "lon"],
pch=22, col="blue", cex=2)
legend(x="topright", legend=c("closest", "Mahalanobis", "SD"), pch=c(22, 3, 1),
col=c("blue", "black", "red"))

analogue.file <- paste(getwd(), '//ensembles//analogue//FloridaSD_south_analogue.grd', sep="")
plot(raster(analogue.file), main="Climatic distance normalized by standard deviation")
points(Florida.analogue.mahal[3:50, "lat"] ~ Florida.analogue.mahal[3:50, "lon"],
pch=3, col="black", cex=1)
points(Florida.analogue.sd[3:50, "lat"] ~ Florida.analogue.sd[3:50, "lon"],
pch=1, col="red", cex=1)
points(Florida.analogue.sd[2, "lat"] ~ Florida.analogue.sd[2, "lon"],
pch=22, col="blue", cex=2)
legend(x="topright", legend=c("closest", "Mahalanobis", "SD"), pch=c(22, 3, 1),
col=c("blue", "black", "red"))
par(mfrow=c(1,1))

## End(Not run)
**Description**

The main function allows for batch processing of different species and different environmental RasterStacks. The function makes internal calls to `ensemble.calibrate.weights`, `ensemble.calibrate.models` and `ensemble.raster`.

**Usage**

```r
ensemble.batch(x = NULL, xn = c(x),
               species.presence = NULL, species.absence = NULL,
               presence.min = 20, thin.km = 0.1,
               an = 1000, excludep = FALSE, target.groups = FALSE,
               get.block = FALSE, block.default = runif(1) > 0.5, get.subblocks = FALSE,
               SSB.reduce = FALSE, CIRCLES.d = 250000,
               k.splits = 4, k.test = 0,
               n.ensembles = 1,
               VIF.max = 10, VIF.keep = NULL,
               SINK = FALSE, CATCH.OFF = 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,
               target.PresenceAbsence = FALSE,
               ENSEMBLE.best = 0, ENSEMBLE.min = 0.7, ENSEMBLE.exponent = 1,
               ENSEMBLE.weight.min = 0.05,
               input.weights = NULL,
               MAXENT = 1, MAXNET = 1, MAXLIKE = 1, GBM = 1, GBMSTEP = 0, RF = 1, CF = 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, GLMMET = 1,
               BIOCLIM.O = 0, BIOCLIM = 1, DOMAIN = 1, MAHAL = 1, MAHAL01 = 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.path = paste(getwd(), "/models/maxent", sep=""),
               MAXNET.classes = "default", MAXNET.clamp = FALSE, MAXNET.type = "cloglog",
               MAXLIKE.formula = NULL, MAXLIKE.method = "BFGS",
               GBM.formula = NULL, GBM.n.trees = 2001,
               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))),
               CF.formula = NULL, CF.ntree = 751, CF.mtry = floor(sqrt(raster::nlayers(x))),
               GLM.formula = NULL, GLM.family = binomial(link = "logit"),
               GLMSTEP.steps = 10000, STEP.formula = NULL, GLMSTEP.scope = NULL, GLMSTEP.k = 2,
               GAM.formula = NULL, GAM.family = binomial(link = "logit"),
               GAMSTEP.steps = 10000, GAMSTEP.scope = NULL, GAMSTEP.pos = 1,
               MGCV.formula = NULL, MGCVFIX.formula = NULL,
```
ensemble.batch

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,
GLMNET.nlambda = 100, GLMNET.class = FALSE,
BIOCLIM.O.fraction = 0.9,
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,
abs.breaks = 6, pres.breaks = 6, sd.breaks = 9,
p = NULL, a = NULL,
pt = NULL, at = NULL,
threshold = -1,
threshold.method = "spec_sens", threshold.sensitivity = 0.9,
threshold.PresenceAbsence = FALSE)

df
ensemble.plot(RASTER.species.name = "Species001", RASTER.stack.name = "base",
plot.method=c("suitability", "presence", "count",
"consensussuitability", "consensuspresence", "consensuscount", "consensussd"),
dev.new.width = 7, dev.new.height = 7,
m = paste(RASTER.species.name, " ", plot.method,
" for ", RASTER.stack.name, sep=""),
positive.filters = c("grd"), negative.filters = c("xml"),
p=NULL, a=NULL,
threshold = -1,
threshold.method = "spec_sens", threshold.sensitivity = 0.9,
threshold.PresenceAbsence = FALSE,
abs.breaks = 6, abs.col = NULL,
pres.breaks = 6, pres.col = NULL,
sd.breaks = 9, sd.col = NULL,
absencePresence.col = NULL,
count.col = NULL,
maptools.boundaries = TRUE, maptools.col = "dimgrey", ...)

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.
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).

**thin.km**

Threshold for minimum distance (km) between presence point locations for focal species for model calibrations in each run. A new data set is randomly selected via `ensemble.spatialThin` in each of ensemble run.

**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`

**target.groups**

Parameter that indicates (if TRUE) that the provided background points (argument `a`) represent presence points from a target group sensu Phillips et al. 2009 (these are species that are all collected or observed using the same methods or equipment). Setting the parameter to TRUE results in selecting the centres of cells of the target groups as background points, while avoiding to select the same cells twice. Via argument `excludep`, it is possible to filter out cells with presence observations (argument `p`).

**get.block**

if TRUE, instead of creating k-fold cross-validation subsets randomly (`kfold`), create 4 subsets of presence and background locations with `get.block`.

**block.default**

if FALSE, instead of making the first division of presence point locations along the y-coordinates (latitude) as in `get.block`, make the first division along the x-coordinates (longitude).

**get.subblocks**

if TRUE, then 4 subsets of presence and background locations are generated in a checkerboard configuration by applying `get.block` to each of the 4 blocks generated by `get.block` in a first step.

**SSB.reduce**

If TRUE, then new background points that will be used for evaluation the suitability models will be selected (`randomPoints`) in circular neighbourhoods (created with `circles`) around presence locations (p and pt). The abbreviation of SSB refers to spatial sorting bias; see also `ssb`.

**CIRCLES.d**

Radius in m of circular neighbourhoods (created with `circles`) around presence locations (p and pt).

**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.calibrate.weights` 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.

VIF.max Maximum Variance Inflation Factor of variables; see ensemble.VIF.

VIF.keep character vector with names of the variables to be kept; see ensemble.VIF.

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.

CATCH.OFF Disable calls to function tryCatch.

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 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.

threshold.method Method to calculate the threshold between predicted absence and presence: possi-
bilities include spec_sens (highest sum of the true positive rate and the true
negative rate), kappa (highest kappa value), no_omission (highest threshold
that corresponds to no omission), prevalence (modeled prevalence is closest to
observed prevalence) and equal_sens_spec (equal true positive rate and true
negative rate). See threshold. Options specific to the BiodiversityR imple-
mentation are: threshold.mean (resulting in calculating the mean value of
spec_sens, equal_sens_spec and prevalence) and threshold.min (result-
ing in calculating the minimum value of spec_sens, equal_sens_spec and
prevalence).

threshold.sensitivity Sensitivity value for threshold.method = 'sensitivity'. See threshold.

threshold.PresenceAbsence If TRUE calculate thresholds with the PresenceAbsence package. See optimal.thresholds.

ENSEMBLE.best The number of individual suitability models to be used in the consensus suitabil-
ity 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.
The minimum input weight (typically corresponding to AUC values) 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.

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.

The minimum output weight for models included in the ensemble, applying to weights that sum to one. Note that `ENSEMBLE.min` typically refers to input AUC values.

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.calibrate.weights` can be used.

Input weight for a maximum entropy model (`maxent`). (Only weights > 0 will be used.)

Number: if larger than 0, then a maximum entropy model (`maxnet`) will be fitted among ensemble

Input weight for a maxlike model (`maxlike`). (Only weights > 0 will be used.)

Input weight for a boosted regression trees model (`gbm`). (Only weights > 0 will be used.)

Input weight for a stepwise boosted regression trees model (`gbm.step`). (Only weights > 0 will be used.)

Input weight for a random forest model (`randomForest`). (Only weights > 0 will be used.)

Number: if larger than 0, then a random forest model (`cforest`) will be fitted among ensemble

Input weight for a generalized linear model (`glm`). (Only weights > 0 will be used.)

Input weight for a stepwise generalized linear model (`stepAIC`). (Only weights > 0 will be used.)

Input weight for a generalized additive model (`gam`). (Only weights > 0 will be used.)

Input weight for a stepwise generalized additive model (`step.gam`). (Only weights > 0 will be used.)

Input weight for a multivariate adaptive regression spline model (`earth`). (Only weights > 0 will be used.)

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.)

GLMNET Input weight for a GLM with lasso or elasticnet regularization (glmnet). (Only weights > 0 will be used.)

BIOCLIM.O Input weight for the original BIOCLIM algorithm (ensemble.bioclim). (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.)

MAHAL01 Input weight for the Mahalobis algorithm (mahal), using a transformation method afterwards whereby output is within the range between 0 and 1. (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 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.

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. 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.
MAXENT.an number of background points for calibration to be selected with randomPoints in case argument MAXENT.a is missing. When used with the ensemble.batch 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.

MAXENT.path path to the directory where output files of the maximum entropy model are stored; see also maxent

MAXNET.classes continuous feature classes, either "default" or any subset of "lqpht" (linear, quadratic, product, hinge, threshold). Note that the "default" option chooses feature classes based on the number of presence locations as "l" (< 10 locations), "lq" (10 - 14 locations), "lqh" (15 - 79 locations) or "lqph" (> 79 locations). See also maxnet.

MAXNET.clamp restrict predictors and features to the range seen during model training; see also predict.maxnet

MAXNET.type type of response required; see also predict.maxnet

MAXLIKE.formula formula for the maxlike algorithm; see also maxlike

MAXLIKE.method method for the maxlike algorithm; see also optim

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.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 the 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

CF.formula formula for random forest algorithm; see also cforest

CF.ntree number of trees to grow in a forest; see also cforest_control

CF.mtry number of input variables randomly sampled as candidates at each node for random forest like algorithms; see also cforest_control

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 stepwise search in 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
GLMNET.nlambd The number of lambda values; see also glmnet
GLMNET.class Use the predicted class to calculate the mean predictions of GLMNET; see also predict.glmnet
BIOCLIM.O.fraction Fraction of range representing the optimal limits, default value of 0.9 as in the original BIOCLIM software (ensemble.bioclim).
MAHAL.shape parameter that influences the transformation of output values of mahal.
RASTER.species.name
First part of the names of the raster files, expected to identify the modelled species (or organism).

RASTER.stack.name
Last part of the names of the raster files, expected to identify the predictor stack used.

positive.filters
vector that indicates parts of filenames for files that will be included in the calculation of the mean probability values

negative.filters
vector that indicates parts of filenames for files that will not be included in the calculation of the mean probability values

abs.breaks
Number of breaks in the colouring scheme for absence (only applies to suitability mapping).

pres.breaks
Number of breaks in the colouring scheme for presence (only applies to suitability mapping).

sd.breaks
Number of breaks in the colouring scheme for standard deviation (only applies to sd mapping).

p
presence points used for calibrating the suitability models, typically available in 2-column (x, y) or (lon, lat) dataframe; see also prepareData and extract

a
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

threshold
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.

plot.method
Choice of maps to be plotted: suitability plots suitability maps, presence plots presence-absence maps, count plots count maps (count of number of algorithms or number of ensembles predicting presence) and sd plots standard deviation maps.

dev.new.width
Width for new graphics device (dev.new). If < 0, then no new graphics device is opened.

dev.new.height
Heigth for new graphics device (dev.new). If < 0, then no new graphics device is opened.

main
main title for the plots.

abs.col
specify colours for absence (see examples on how not to plot areas where the species is predicted absent)

pres.col
specify colours for presence

sd.col
specify colours for standard deviation
absencePresence.col
 specify colours for absence - presence maps (see examples on how not to plot
 areas where the species is predicted absent)

count.col
 specify colours for number of algorithms or ensembles (see examples on how
 not to plot areas where the species is predicted absent)

maptools.boundaries
 If TRUE, then plot approximate country boundaries wrld_simpl

maptools.col
 Colour for approximate country boundaries plotted via wrld_simpl

... Other items passed to function plot.

Details

This function allows for batch processing of different species and different environmental Raster-
Stacks. The function makes internal calls to ensemble.spatialThin, ensemble.VIF, ensemble.calibrate.weights,
ensemble.calibrate.models and ensemble.raster.

Different ensemble runs allow for different random selection of k-fold subsets, background locations
or spatial thinning of presence locations.

ensemble.calibrate.weights 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 (in-
cluding the possibility for weights = 0).

ensemble.calibrate.models 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). After the mean
probability has been calculated, the niche overlap (nicheOverlap) with the different input layers is
calculated.

Function ensemble.plot plots suitability, presence-absence or count maps. In the case of suit-
ability maps, the presence-absence threshold needs to be provide as suitabilities smaller than the
threshold will be coloured red to orange, whereas suitabilities larger than the threshold will be
coloured light blue to dark blue.

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


See Also

ensemble.calibrate.weights, ensemble.calibrate.models, 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
background <- randomPoints(predictors, n=1000, extf = 1.00)

# north and south for new predictions (as if new climates)
ext2 <- extent(-90, -32, 0, 23)
predictors2 <- crop(predictors, y=ext2)
predictors2 <- stack(predictors2)
predictors2@title <- "north"

ext3 <- extent(-90, -32, -33, 0)
predictors3 <- crop(predictors, y=ext3)
predictors3 <- stack(predictors3)
predictors3@title <- "south"

# fit 3 ensembles with batch processing, choosing the best ensemble model based on the
# average weights of 4-fold split of calibration and testing data
# final models use all available presence data and average weights determined by the
# ensemble.callibrate.weights 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,
  xn=c(predictors, predictors2, predictors3),
  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, 3),
  ENSEMBLE.min=0.7,
  MAXENT=0, MAXNET=1, MAXLIKE=1, GBM=1, GBMSTEP=0, RF=1, CF=1,
  GLM=1, GLMSTEP=1, GAM=1, GAMSTEP=1, MGCV=1, MGCVFIX=1,
  EARTH=1, RPART=1, NNET=1, SVM=1, SVM=1, GLMM=1, BIOCLIM=1,
  BIOCLIM.O=1, BIOCLIM=1, DOMAIN=1, MAHAL=0, MAHAL01=1,
  PROBIT=TRUE,
  Yweights="BIOMOD",
  formulae.defaults=TRUE)

# summaries for the 3 ensembles for the species
# summaries are based on files in folders ensemble/suitability,
# ensemble/presence and ensemble/count
# ensemble.mean is used internally in ensemble.batch

ensemble.mean(RASTER.species.name="Bradypus", RASTER.stack.name="base",
  p=pres, a=background)

# plot mean suitability without specifying colours
plot1 <- ensemble.plot(RASTER.species.name="Bradypus", RASTER.stack.name="base",
  plot.method="consensussuitability",
  p=pres, a=background, abs.breaks=4, pres.breaks=9)
plot1

# only colour the areas where species is predicted to be present
# option is invoked by having no absence breaks
# same colourscheme as \url{http://www.worldagroforestry.org/atlas-central-america}
LAatlascols <- grDevices::colorRampPalette(c("#FFFF80", "#38E009", "#1A93AB", "#0C1078"))
plot2 <- ensemble.plot(RASTER.species.name="Bradypus", RASTER.stack.name="base",
  plot.method="consensussuitability",
  p=pres, a=background, abs.breaks=0, pres.breaks=9, pres.col=LAatlascols(8))
plot2

# only colour the areas where species is predicted to be present
# option is invoked by only setting one colour for absence-presence
plot3 <- ensemble.plot(RASTER.species.name="Bradypus", RASTER.stack.name="base",
  plot.method="consensuspresence",
  absencePresence.col=c("#90EE90"))

# only colour presence area by specifying colours > 0
plot4 <- ensemble.plot(RASTER.species.name="Bradypus", RASTER.stack.name="base", plot.method="consensuscount", count.col=LAatlascols(3))

## End(Not run)

**ensemble.bioclim**  
*Suitability mapping based on the BIOCLIM algorithm*

**Description**

Implementation of the BIOCLIM algorithm more similar to the original BIOCLIM algorithm and software than the implementation through `bioclim`. Function `ensemble.bioclim` creates the suitability map. Function `ensemble.bioclim.object` provides the reference values used by the prediction function used by `predict`.

**Usage**

```r
ensemble.bioclim(x = NULL, bioclim.object = NULL, RASTER.object.name = bioclim.object$species.name, RASTER.stack.name = x@title, RASTER.format = "raster", KML.out = TRUE, KML.blur = 10, KML.maxpixels = 100000, CATCH.OFF = FALSE)

ensemble.bioclim.object(x = NULL, p = NULL, fraction = 0.9, quantiles = TRUE, species.name = "Species001", factors = NULL)
```

**Arguments**

- **x**
  RasterStack object (stack) containing all environmental layers for which suitability should be calculated, or alternatively a data.frame containing the bioclimatic variables.

- **bioclim.object**
  Object listing optimal and absolute minima and maxima for bioclimatic variables, used by the prediction function that is used internally by `predict`. This object is created with `ensemble.bioclim.object`.

- **RASTER.object.name**
  First part of the names of the raster file that will be generated, expected to identify the species or crop for which ranges were calculated.

- **RASTER.stack.name**
  Last part of the names of the raster file that will be generated, expected to identify the predictor stack used.
ensemble.bioclim

**RASTER.format** Format of the raster files that will be generated. See `writeFormats` and `writeRaster`.

**KML.out** If TRUE, then kml files will be saved in a subfolder 'kml/zones'.

**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.

**KML.maxpixels** Maximum number of pixels for the PNG image that will be displayed in Google Earth. See also KML.

**CATCH.OFF** Disable calls to function `tryCatch`.

**p** presence points used for calibrating the suitability models, typically available in 2-column (lon, lat) dataframe; see also `prepareData` and `extract`.

**fraction** Fraction of range representing the optimal limits, default value of 0.9 as in the original BIOCLIM software.

**quantiles** If TRUE then optimal limits are calculated as quantiles corresponding to 0.5-fraction/2 and 0.5+fraction/2 percentiles. If FALSE then optimal limits are calculated from the normal distribution with mean -cutoff*sd and mean + cutoff*sd with cutoff calculated as `qnorm(0.5+fraction/2)`.

**species.name** Name by which the model results will be saved.

**factors** vector that indicates which variables are factors; these variables will be ignored by the BIOCLIM algorithm

**Details**

Function `ensemble.bioclim` maps suitability for a species based on optimal (percentiles, typically 5 and 95 percent) and absolute (minimum to maximum) limits for bioclimatic variables. If all values at a given location are within the optimal limits, suitability values are mapped as 1 (suitable). If not all values are within the optimal limits, but all values are within the absolute limits, suitability values are mapped as 0.5 (marginal). If not all values are within the absolute limits, suitability values are mapped as 0 (unsuitable).

Function `ensemble.bioclim.object` calculates the optimal and absolute limits. Optimal limits are calculated based on the parameter `fraction`, resulting in optimal limits that correspond to 0.5-fraction/2 and 0.5+fraction/2 (the default value of 0.9 therefore gives a lower limit of 0.05 and a upper limit of 0.95). Two methods are implemented to obtain optimal limits for the lower and upper limits. One method (quantiles = FALSE) uses mean, standard deviation and a cutoff parameter calculated with `qnorm`. The other method (quantiles = TRUE) calculates optimal limits via the `quantile` function. To handle possible asymmetrical distributions better, the second method is used as default.

When `x` is a RasterStack and point locations are provided, then optimal and absolute limits correspond to the bioclimatic values observed for the locations. When `x` is RasterStack and point locations are not provided, then optimal and absolute limits correspond to the bioclimatic values of the RasterStack.

Applying to algorithm without providing point locations will provide results that are similar to the `ensemble.novel` function, whereby areas plotted as not suitable will be the same areas that are novel.
Value

Function `ensemble.bioclim.object` returns a list with following objects:

- `lower.limits` vector with lower limits for each bioclimatic variable
- `upper.limits` vector with upper limits for each bioclimatic variable
- `minima` vector with minima for each bioclimatic variable
- `maxima` vector with maxima for each bioclimatic variable
- `means` vector with mean values for each bioclimatic variable
- `medians` vector with median values for each bioclimatic variable
- `sds` vector with standard deviation values for each bioclimatic variable
- `cutoff` cutoff value for the normal distribution
- `fraction` fraction of values within the optimal limits
- `species.name` name for the species

Author(s)

Roeland Kindt (World Agroforestry Centre) with inputs from Trevor Booth (CSIRO)

References


See Also

`bioclim`, `ensemble.bioclim.graph` and `ensemble.novel`

Examples

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

background <- dismo::randomPoints(predictors, n=100)
colnames(background)=c('lon', 'lat')

pres.dataset <- data.frame(extract(predictors, y=pres))
names(pres.dataset) <- names(predictors)
pres.dataset$biome <- as.factor(pres.dataset$biome)

Bradypus.bioclim <- ensemble.bioclim.object(predictors, quantiles=T,
p=pres, factors="biome", species.name="Bradypus")

Bradypus.bioclim

# obtain the same results with a data.frame
Bradypus.bioclim2 <- ensemble.bioclim.object(pres.dataset, quantiles=T,
  species.name="Bradypus")

Bradypus.bioclim2

# obtain results for entire rasterStack
Bradypus.bioclim3 <- ensemble.bioclim.object(predictors, p=NULL, quantiles=T,
   factors="biome", species.name="America")

Bradypus.bioclim3

ensemble.bioclim(x=predictors, bioclim.object=Bradypus.bioclim, KML.out=T)
ensemble.bioclim(x=predictors, bioclim.object=Bradypus.bioclim3, KML.out=T)

par.old <- graphics::par(no.readonly=T)

classical <- paste("ensembles//Bradypus_base_BIOCLIM_orig", sep="")
raster::plot(raster(classical), breaks=c(-0.1, 0, 0.5, 1),
  col=c("grey", "blue", "green"), main="original method")

classical2 <- paste("ensembles//America_base_BIOCLIM_orig", sep="")
raster::plot(raster(classical2), breaks=c(-0.1, 0, 0.5, 1),
  col=c("grey", "blue", "green"), main="America")

graphics::par(par.old)

# compare with implementation bioclim in dismo
bioclim.dismo <- bioclim(predictors, p=pres)
classical2 <- paste("ensembles//Bradypus_base_BIOCLIM_dismo", sep="")
raster::predict(object=predictors, model=bioclim.dismo, na.rm=TRUE,
  filename=classical2, progress='text', overwrite=TRUE)

par.old <- graphics::par(no.readonly=T)

classical2 <- paste("ensembles//Bradypus_base_BIOCLIM_dismo", sep="")
raster::plot(raster(classical2), breaks=c(-0.1, 0, 0.5, 1),
  col=c("grey", "blue", "green"), main="dismo method")

graphics::par(par.old)

# use dummy variables to deal with factors
```
predictors <- stack(predictor.files)
predictor.files <- list.files(path=paste(system.file(package="dismo"), '/ex', sep=''), pattern='grd', full.names=TRUE)
predictors <- stack(predictor.files)
predictors.dummy <- subset(predictors, subset=c("biome_1", "biome_2", "biome_3", "biome_4", "biome_5", "biome_7", "biome_8", "biome_9", "biome_10", "biome_12", "biome_13", "biome_14"))
predictors.dummy@title <- "base_dummy"

Bradypus.dummy <- ensemble.bioclim.object(predictors.dummy, quantiles=T, p=pres, species.name="Bradypus")

par.old <- graphics::par(no.readonly=T)

rasterfull3 <- paste("ensembles//Bradypus_base_dummy_BIOCLIM_orig", sep="")
raster::plot(raster(rasterfull1), breaks=c(-0.1, 0, 0.5, 1), col=c("grey", "blue", "green"), main="numeric predictors")
raster::plot(raster(rasterfull3), breaks=c(-0.1, 0, 0.5, 1), col=c("grey", "blue", "green"), main="dummy predictors")

## End(Not run)

---

**ensemble.bioclim.graph**

*Graphs of bioclimatic ranges of species and climates*

**Description**

The main graph function makes graphs that show mean, median, minimum, maximum and lower and upper limits for species or climates. The `ensemble.bioclim.graph.data` function creates input data, using `ensemble.bioclim.object` internally.

**Usage**

```r
ensemble.bioclim.graph(graph.data = NULL, focal.var = NULL, 
species.climates.subset = NULL, cols = NULL, 
var.multiply = 1.0, ref.lines = TRUE)
```
ensemble.bioclim.graph.data(
    x=NULL, p=NULL, fraction = 0.9,
    speciesclimate.name="Species001_base", factors = NULL)

Arguments

graph.data  Input data with same variables as created by ensemble.bioclim.graph
focal.var   Bioclimatic variable to be plotted in the graph
species.climates.subset
            Character vector with subset of names of species and climates to be plotted in
            the graph (if not provided, then all species and climates will be plotted).
cols        colours for the different species and climates
var.multiply multiplier for the values to be plotted; 0.1 should be used if the bioclimatic vari-
            able was multiplied by 10 in the raster layers as in WorldClim and AFRICLIM
ref.lines   If TRUE, then horizontal reference lines will be added for the minimum and max-
            imum values of the species or climate plotted on the extreme left in the graph
x           RasterStack object (stack) containing all environmental layers for which statis-
            tics should be calculated; see also ensemble.bioclim.
p           presence points used for calibrating the suitability models, typically available in
            2-column (lon, lat) dataframe; see also ensemble.bioclim.
fraction    Fraction of range representing the optimal limits, default value of 0.9 as in the
            original BIOCLIM software; see also ensemble.bioclim.
species.climate.name
            Name for the species or climate that will be used as label in the graph.
factors     vector that indicates which variables are factors; these variables will be ignored
            by the BIOCLIM algorithm; see also ensemble.bioclim.

Details

The function creates a graph that shows mean, median, minimum, maximum and upper and lower
limits for a range of species and climates. The graph can be useful in interpreting results of
ensemble.bioclim or ensemble.novel.

In the graphs, means are indicated by an asterisk (pch=8 and medians as larger circles (pch=1)).

Value

function ensemble.bioclim.graph.data creates a data frame, function codeensemble.bioclim.graph
allows for plotting.

Author(s)

Roeland Kindt (World Agroforestry Centre)

See Also

ensemble.bioclim
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]

# climates for north and south (use same process for future climates)
ext2 <- extent(-90, -32, 0, 23)
predictors2 <- crop(predictors, y=ext2)
predictors2 <- stack(predictors2)
predictors2@title <- "north"

text3 <- extent(-90, -32, -33, 0)
predictors3 <- crop(predictors, y=ext3)
predictors3 <- stack(predictors3)
predictors3@title <- "south"

graph.data1 <- ensemble.bioclim.graph.data(predictors, p=pres,
    factors="biome", species.climate.name="Bradypus")
graph.data2 <- ensemble.bioclim.graph.data(predictors, p=NULL,
    factors="biome", species.climate.name="baseline")
graph.data3 <- ensemble.bioclim.graph.data(predictors2, p=NULL,
    factors="biome", species.climate.name="north")
graph.data4 <- ensemble.bioclim.graph.data(predictors3, p=NULL,
    factors="biome", species.climate.name="south")
graph.data.all <- rbind(graph.data1, graph.data2, graph.data3, graph.data4)

par.old <- graphics::par(no.readonly=T)
graphics::par(mfrow=c(2, 2))

ensemble.bioclim.graph(graph.data.all, focal.var="bio5",
    var.multiply=0.1, cols=c("black", rep("blue", 3)))
ensemble.bioclim.graph(graph.data.all, focal.var="bio6",
    var.multiply=0.1, cols=c("black", rep("blue", 3)))
ensemble.bioclim.graph(graph.data.all, focal.var="bio16",
    var.multiply=1.0, cols=c("black", rep("blue", 3)))
ensemble.bioclim.graph(graph.data.all, focal.var="bio17",
    var.multiply=1.0, cols=c("black", rep("blue", 3)))

graphics::par(par.old)
ensemble.calibrate.models

Suitability mapping based on ensembles of modelling algorithms: calibration of models and weights

Description

The basic function `ensemble.calibrate.models` 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

```r
ensemble.calibrate.models(x = NULL, p = NULL,
                           a = NULL, an = 1000, excludep = FALSE, target.groups = FALSE,
                           k = 0, pt = NULL, at = NULL, SSB.reduce = FALSE, CIRCLES.d = 250000,
                           TrainData = NULL, TestData = NULL,
                           VIF = FALSE, COR = FALSE,
                           SINK = FALSE, PLOTS = FALSE, CATCH.OFF = FALSE,
                           threshold.method = "spec_sens", threshold.sensitivity = 0.9,
                           threshold.PresenceAbsence = FALSE,
                           evaluations.keep = FALSE,
                           models.list = NULL, models.keep = FALSE,
                           models.save = FALSE, species.name = "Species001",
                           ENSEMBLE.tune = FALSE,
                           ENSEMBLE.best = 0, ENSEMBLE.min = 0.7, ENSEMBLE.exponent = 1,
                           ENSEMBLE.weight.min = 0.05,
                           input.weights = NULL,
                           MAXENT = 1, MAXNET = 1, MAXLIKE = 1, GBM = 1, GBMSTEP = 1, RF = 1, CF = 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, GLMNET = 1,
                           BIOCLIM.O = 0, BIOCLIM = 1, DOMAIN = 1, MAHAL = 1, MAHAL01 = 1,
                           PROBIT = FALSE,
                           Yweights = "BIOMOD",
                           layer.drops = NULL, factors = NULL, dummy.vars = NULL,
```
ensemble.calibrate.models

formulae.defaults = TRUE, maxit = 100,
MAXENT.a = NULL, MAXENT.an = 10000,
MAXENT.path = paste(getwd(), "/models/maxent_{species.name}_{sep}"),
MAXENT.classes = "default", MAXENT.clamp = FALSE, MAXENT.type = "cloglog",
MAXLIKE.formula = NULL, MAXLIKE.method = "BFGS",
GBM.formula = NULL, GBM.n.trees = 2001,
GBMSTEP.gbm.x = 2:(ncol(TrainData.orig)), 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))),
CF.formula = NULL, CF.ntree = 751, CF.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,
GLMNET.nlambda = 100, GLMNET.class = FALSE,
MAHAL.shape = 1)

ensemble.calibrate.weights(x = NULL, p = NULL, TrainTestData=NULL,
a = NULL, an = 1000,
get.block = FALSE, block.default = TRUE, get.subblocks = FALSE,
SSB.reduce = FALSE, CIRCLES.d = 100000,
excludep = FALSE, target.groups = FALSE,
k = 4,
VIF = FALSE, COR = FALSE,
SINK = FALSE, PLOTS = FALSE, CATCH.OFF = FALSE,
data.keep = FALSE,
species.name = "Species001",
threshold.method = "spec_sens", threshold.sensitivity = 0.9,
threshold.PresenceAbsence = FALSE,
ENSEMBLE.tune = FALSE,
ENSEMBLE.best = 0, ENSEMBLE.min = 0.7, ENSEMBLE.exponent = 1,
ENSEMBLE.weight.min = 0.05,
input.weights = NULL,
MAXENT = 1, MAXNET = 1, MAXLIKE = 1, GBM = 1, GBMSTEP = 1, RF = 1, CF = 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, GLMNET = 1,
ensemble.calibrate.models

BIOCLIM.O = 0, BIOCLIM = 1, DOMAIN = 1, MAHAL = 1, MAHAL01 = 1,
PROBIT = FALSE,
Yweights = "BIOMOD",
layers.drops = NULL, factors = NULL, dummy.vars = NULL,
formulae.defaults = TRUE, maxit = 100,
MAXENT.a = NULL, MAXENT.an = 10000,
MAXENT.path = paste(getwd(), "/models/maxent\_", species.name, sep=""),
MAXNET.classes = "default", MAXNET.clamp = FALSE, MAXNET.type = "cloglog",
MAXLIKE.formula = NULL, MAXLIKE.method = "BFGS",

GBM.formula = NULL, GBM.n.trees = 2001,
GBMSTEP.gbm.x = 2:length(var.names)+1, GBMSTEP.tree.complexity = 5,

GBMNSTEP.learning.rate = 0.005,
GBMNSTEP.bag.fraction = 0.5, GBMNSTEP.step.size = 100,
RF.formula = NULL, RF.ntree = 751, RF.mtry = floor(sqrt(length(var.names)))),
CF.formula = NULL, CF.ntree = 751, CF.mtry = floor(sqrt(length(var.names)))),
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),
RPAR.T.formula = NULL, RPAR.T.xval = 50,
NNET.formula = NULL, NNET.size = 8, NNET.decay = 0.01,
FDA.formula = NULL,
SVR.formula = NULL,
SVME.formula = NULL,
GLMNET.nlambda = 100, GLMNET.class = FALSE,
BIOCLIM.O.fraction = 0.9,
MAHAL.shape = 1)

ensemble.calibrate.models.gbm(x = NULL, p = NULL, a = NULL, an = 1000, excludep = FALSE,

k = 4,
TrainData = NULL,
VIF = FALSE, COR = FALSE,
SINK = FALSE, PLOTS = FALSE,
 species.name = "Species001",
Yweights = "BIOMOD",
layers.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.calibrate.models.nnet(x = NULL, p = NULL, a = NULL, an = 1000, excludep = FALSE,

k = 4,
TrainData = NULL,
VIF = FALSE, COR = FALSE,
ensemble.calibrate.models

SINK = FALSE, PLOTS = FALSE,
species.name = "Species001",
Yweights = "BIOMOD",
layer.drops = NULL, factors = NULL,
formulaedefaults = 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, target.groups = FALSE,
k = 0, pt = NULL, at = NULL, SSB.reduce = FALSE, CIRCLES.d = 100000,
TrainData = NULL, TestData = NULL,
VIF = FALSE, COR = FALSE,
SINK = FALSE,
species.name = "Species001",
difference = FALSE, variables.alone = FALSE,
ENSEMBLE.tune = FALSE,
ENSEMBLE.best = 0, ENSEMBLE.min = 0.7, ENSEMBLE.exponent = 1,
input.weights = NULL,
MAXENT = 1, MAXNET = 1, MAXLIKE = 1, GBM = 1, GBMSTEP = 0, RF = 1, CF = 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, GLMNET = 1,
BIOLUM.0 = 0, BIOLUM = 1, DOMAIN = 1, MAHAL = 1, MAHAL.01 = 1,
PROBIT = FALSE,
Yweights = "BIOMOD",
layer.drops = NULL, factors = NULL, dummy.vars = NULL,
maxit = 100,
MAXENT.a = NULL, MAXENT.an = 10000,
MAXENT.path = paste(getwd(), "/models/maxent\_", species.name, sep=""),
MAXNET.classes = "default", MAXNET.clamp = FALSE, MAXNET.type = "cloglog",
MAXLIKE.method = "BFGS",
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,
CF.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,
GLMNET.nlambda = 100, GLMNET.class = FALSE,
BIOLUM.0.fraction = 0.9,
MAHAL.shape = 1)
ensemble.weights(weights = c(0.9, 0.8, 0.7, 0.5),
    best = 0, min.weight = 0,
    exponent = 1, digits = 6)

ensemble.strategy(TrainData = NULL, TestData = NULL,
    verbose = FALSE,
    ENSEMBLE.best = c(4:10), ENSEMBLE.min = c(0.7),
    ENSEMBLE.exponent = c(1, 2, 3) )

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

ensemble.threshold(eval, threshold.method = "spec_sens", threshold.sensitivity = 0.9,
    threshold.PresenceAbsence = FALSE, Pres, Abs)

ensemble.VIF(x = NULL, a = NULL, an = 10000,
    layer.drops = NULL, factors = NULL, dummy.vars = NULL,
    VIF.max = 10, keep = NULL)

ensemble.VIF.dataframe(x=NULL, VIF.max=10, car=TRUE, silent=F)

ensemble.pairs(x = NULL, a = NULL, an = 10000)

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

excludelp parameter that indicates (if TRUE) that presence points will be excluded from the
background points; see also randomPoints

target.groups Parameter that indicates (if TRUE) that the provided background points (argument
a) represent presence points from a target group sensu Phillips et al. 2009 (these
are species that are all collected or observed using the same methods or equip-
ment). Setting the parameter to TRUE results in selecting the centres of cells of
the target groups as background points, while avoiding to select the same cells
twice. Via argument excludelp, it is possible to filter out cells with presence
observations (argument p).

k If larger than 1, the number of groups to split between calibration (k-1) and eval-
uation (1) data sets (for example, k = 4 results in 3/4 of presence and background
points to be used for calibrating the models, and 1/4 of presence and background points to be used for evaluating the models). For ensemble.calibrate.weights, ensemble.calibrate.models.gbm and ensemble.calibrate.models.nnet, 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

SSB.reduce

If TRUE, then new background points that will be used for evaluatings the suitability models will be selected (randomPoints) in circular neighbourhoods (created with circles) around presence locations (p and pt). The abbreviation of SSB refers to spatial sorting bias; see also ssb.

CIRCLES.d

Radius in m of circular neighbourhoods (created with circles) around presence locations (p and pt).

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 MAXENT.TrainData.

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 ensemble.strategy, this dataframe should be a dataframe that contains predictions for various models - such dataframe can be provided by the ensemble.calibrate.models 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 species.name. In case the file already exists, then results are appended. See also sink.

PLOTS

Disabled option of plotting evaluation results (BiodiversityR version 2.9-1) - see examples how to plot results afterwards and also evaluate.

CATCH.OFF

Disable calls to function tryCatch.

threshold.method

Method to calculate the threshold between predicted absence and presence; possibilities include spec_sens (highest sum of the true positive rate and the true negative rate), kappa (highest kappa value), no_omission (highest threshold that corresponds to no omission), prevalence (modeled prevalence is closest to observed prevalence) and equal_sens_spec (equal true positive rate and true negative rate). See threshold. Options specific to the BiodiversityR implementation are: threshold2005.mean, threshold2005.min, threshold2013.mean and threshold2013.min (resulting in calculating the mean or minimum value
of recommended threshold values by studies published in 2005 and 2013; see details below).

**threshold.sensitivity**

Sensitivity value for threshold.method = 'sensitivity'. See `threshold`.

**threshold.PresenceAbsence**

If TRUE calculate thresholds with the PresenceAbsence package. See `optimal.thresholds`.

**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).

**ENSEMBLE.tune**

Determine weights for the ensemble model based on AUC values (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 (typically corresponding to AUC values) 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.

**ENSEMBLE.weight.min**

The minimum output weight for models included in the ensemble, applying to weights that sum to one. Note that `ENSEMBLE.min` typically refers to input AUC values.

**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.calibrate.weights` can be used.

**MAXENT**

number: if larger than 0, then a maximum entropy model (**maxent**) will be fitted among ensemble

**MAXNET**

number: if larger than 0, then a maximum entropy model (**maxnet**) will be fitted among ensemble
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAXLIKE</td>
<td>Number: if larger than 0, then a maxlike model (<code>maxlike</code>) will be fitted among ensemble</td>
</tr>
<tr>
<td>GBM</td>
<td>Number: if larger than 0, then a boosted regression trees model (<code>gbm</code>) will be fitted among ensemble</td>
</tr>
<tr>
<td>GBMSTEP</td>
<td>Number: if larger than 0, then a stepwise boosted regression trees model (<code>gbm.step</code>) will be fitted among ensemble</td>
</tr>
<tr>
<td>RF</td>
<td>Number: if larger than 0, then a random forest model (<code>randomForest</code>) will be fitted among ensemble</td>
</tr>
<tr>
<td>CF</td>
<td>Number: if larger than 0, then a random forest model (<code>cforest</code>) will be fitted among ensemble</td>
</tr>
<tr>
<td>GLM</td>
<td>Number: if larger than 0, then a generalized linear model (<code>glm</code>) will be fitted among ensemble</td>
</tr>
<tr>
<td>GLMSTEP</td>
<td>Number: if larger than 0, then a stepwise generalized linear model (<code>stepAIC</code>) will be fitted among ensemble</td>
</tr>
<tr>
<td>GAM</td>
<td>Number: if larger than 0, then a generalized additive model (<code>gam</code>) will be fitted among ensemble</td>
</tr>
<tr>
<td>GAMSTEP</td>
<td>Number: if larger than 0, then a stepwise generalized additive model (<code>step.gam</code>) will be fitted among ensemble</td>
</tr>
<tr>
<td>MGCV</td>
<td>Number: if larger than 0, then a generalized additive model (<code>gam</code>) will be fitted among ensemble</td>
</tr>
<tr>
<td>MGCVFIX</td>
<td>Number: if larger than 0, then a generalized additive model with fixed d.f. regression splines (<code>gam</code>) will be fitted among ensemble</td>
</tr>
<tr>
<td>EARTH</td>
<td>Number: if larger than 0, then a multivariate adaptive regression spline model (<code>earth</code>) will be fitted among ensemble</td>
</tr>
<tr>
<td>RPART</td>
<td>Number: if larger than 0, then a recursive partitioning and regression tree model (<code>rpart</code>) will be fitted among ensemble</td>
</tr>
<tr>
<td>NNET</td>
<td>Number: if larger than 0, then an artificial neural network model (<code>nnet</code>) will be fitted among ensemble</td>
</tr>
<tr>
<td>FDA</td>
<td>Number: if larger than 0, then a flexible discriminant analysis model (<code>fda</code>) will be fitted among ensemble</td>
</tr>
<tr>
<td>SVM</td>
<td>Number: if larger than 0, then a support vector machine model (<code>ksvm</code>) will be fitted among ensemble</td>
</tr>
<tr>
<td>SVME</td>
<td>Number: if larger than 0, then a support vector machine model (<code>svm</code>) will be fitted among ensemble</td>
</tr>
<tr>
<td>GLMNET</td>
<td>Number: if larger than 0, then a GLM with lasso or elasticnet regularization (<code>glmnet</code>) will be fitted among ensemble</td>
</tr>
<tr>
<td>BIOCLIM.0</td>
<td>Number: if larger than 0, then the original BIOCLIM algorithm (<code>ensemble.bioclim</code>) will be fitted among ensemble</td>
</tr>
<tr>
<td>BIOCLIM</td>
<td>Number: if larger than 0, then the BIOCLIM algorithm (<code>bioclim</code>) will be fitted among ensemble</td>
</tr>
<tr>
<td>DOMAIN</td>
<td>Number: if larger than 0, then the DOMAIN algorithm (<code>domain</code>) will be fitted among ensemble</td>
</tr>
</tbody>
</table>
ensemble.calibrate.models

MAHAL
number: if larger than 0, then the Mahalanobis algorithm (mahal) will be fitted among ensemble

MAHAL01
number: if larger than 0, then the Mahalanobis algorithm (mahal) will be fitted among ensemble, using a transformation method afterwards whereby output is within the range between 0 and 1 (see details)

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.

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.

MAXENT.an
number of background points for calibration to be selected with randomPoints in case argument MAXENT.a is missing

MAXENT.path
path to the directory where output files of the maximum entropy model are stored; see also maxent

MAXNET.classes
continuous feature classes, either "default" or any subset of "lqph" (linear, quadratic, product, hinge, threshold). Note that the "default" option chooses feature classes based on the number of presence locations as "l" (< 10 locations), "lq" (10 - 14 locations), "lqh" (15 - 79 locations) or "lqph" (> 79 locations). See also maxnet.

MAXNET.clamp
restrict predictors and features to the range seen during model training; see also predict.maxnet

MAXNET.type
type of response required; see also predict.maxnet

MAXLIKE.formula
formula for the maxlike algorithm; see also maxlike

MAXLIKE.method
method for the maxlike algorithm; see also optim

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

CF.formula  formula for random forest algorithm; see also cforest

CF.ntree  number of trees to grow in a forest; see also cforest_control

CF.mtry  number of input variables randomly sampled as candidates at each node for random forest like algorithms; see also cforest_control

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 stepwise search; see also step.gam

GAMSTEP.pos  parameter expected to be set to 1 to allow for fitting of the stepwise generalized additive model
ensemble.calibrate.models

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

GLMNET.nlambda The number of lambda values; see also glmnet

GLMNET.class Use the predicted class to calculate the mean predictions of GLMNET; see predict.glmnet

BIOCLIM.O.fraction Fraction of range representing the optimal limits, default value of 0.9 as in the original BIOCLIM software (ensemble.bioclim).

MAHAL.shape parameter that influences the transformation of output values of mahal. See details section.

TrainTestData 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. This data set will also be used for the maximum entropy and maximum likelihood models.

get.block if TRUE, instead of creating k-fold cross-validation subsets randomly (kfold), create 4 subsets of presence and background locations with get.block.

block.default if FALSE, instead of making the first division of presence point locations along the y-coordinates (latitude) as in get.block, make the first division along the x-coordinates (longitude).

get.subblocks if TRUE, then 4 subsets of presence and background locations are generated in a checkerboard configuration by applying get.block to each of the 4 blocks generated by get.block in a first step.

complexity vector with values of complexity of individual trees (tree.complexity) for boosted regression trees; see also gbm.step

learning vector with values of weights applied to individual trees (learning.rate) for boosted regression trees; see also gbm.step
ensemble.calibrate.models

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.

variables.alone if TRUE, then models are also fitted using each explanatory variable as single explanatory variable

weights input weights for the ensemble.weights 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 ensemble.strategy

eval ModelEvaluation object obtained by evaluate

Pres Suitabilities (probabilities) at presence locations

Abs Suitabilities (probabilities) at background locations

VIF.max Maximum Variance Inflation Factor of the selected subset of variables. In case that at least one of the variables has VIF larger than VIF.max, then the variable with the highest VIF will be removed in the next step.

keep character vector with names of the variables to be kept.

car Also provide results from vif.

silent Limit textual output.

Details

The basic function ensemble.calibrate.models 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 can be 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).
Output from the GLMNET algorithm is calculated as the mean of the output from `predict.glmnet`. With option `GLMNET.class = TRUE`, the mean output is the mean prediction of class 1. With option `GLMNET.class = FALSE`, the mean output is the mean of the responses.

As the Mahalanobis function (`mahal`) does not always provide values within the range of 0 - 1, the output values are rescaled with option `MAHAL01` 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 and the PROBIT transformation may help here (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. In case that `ENSEMBLE.tune` is set to `FALSE`, then the ensemble is calculated based on the input weights.

Function `ensemble.calibrate.weights` 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, $k = 4$ results in splitting the locations in 4 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.calibrate.weights` 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.calibrate.models.gbm` allows to test various combinations of parameters `tree.complexity` and `learning.rate` for the `gbm.step` model.

Function `ensemble.calibrate.models.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 $-2 \times (\text{sum}(x \times \log(x)) + \text{sum}((1-x) \times \log(1-x)))$ 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.calibrate.models and ensemble.raster. This function is always used internally by the ensemble.drop1 function.

The ensemble.weights function is used internally by the ensemble.calibrate.models 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.calibrate.models function, 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.

The ensemble.threshold function is used internally by the ensemble.calibrate.models, ensemble.mean and ensemble.plot functions. threshold2005.mean results in calculating the mean value of threshold methods that resulted in better results (calculated by optimal.thresholds with methods of ObsPrev, MeanProb, MaxSens+Spec, Sens=Spec and MinROCdist) in a study by Liu et al. (Ecography 28: 385-393. 2005). threshold2005.min results in calculating the mean value of threshold methods that resulted in better results (calculated by optimal.thresholds with methods of ObsPrev, MeanProb and MaxSens+Spec) in a study by Liu et al. (Ecography 28: 385-393. 2005). threshold2013.mean results in calculating the mean value of threshold methods that resulted in better results (calculated by optimal.thresholds with methods of ObsPrev, MeanProb, MaxSens+Spec, Sens=Spec and MinROCdist) in a study by Liu et al. (J. Biogeogr. 40: 778-789. 2013). threshold2013.min results in calculating the minimum value of threshold methods that resulted in better results (calculated by optimal.thresholds with methods of ObsPrev, MeanProb, MaxSens+Spec, Sens=Spec and MinROCdist) in a study by Liu et al. (J. Biogeogr. 40: 778-789. 2013).

Function ensemble.VIF implements a stepwise procedure whereby the explanatory variable with highest Variance Inflation Factor is removed from the list of variables. The procedure ends when no variable has VIF larger than parameter VIF.max.

Function ensemble.pairs provides a matrix of scatterplots similar to the example of pairs for version 3.4.3 of that package.

Value

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

Function ensemble.calibrate.weights 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.calibrate.models.gbm and ensemble.calibrate.models.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


See Also

ensemble.raster, ensemble.batch

Examples

```r
## 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 <- "predictors"

# 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 (1/4) and training (3/4) data
groupp <- kfold(pres, 4)
pres_train <- pres[groupp != 1, ]
pres_test <- pres[groupp == 1, ]

# choose background points
background <- randomPoints(predictors, n=1000, extf=1.00)
colnames(background)=c('lon', 'lat')
groupa <- kfold(background, 4)
backg_train <- background[groupa != 1, ]
backg_test <- background[groupa == 1, ]
```
ensemble.calibrate.models

# 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)
# factors removed for BIOCLIM, DOMAIN, MAHAL
ensemble.nofactors <- ensemble.calibrate.models(x=predictors, p=pres_train, a=backg_train,
pt=pres_test, at=backg_test,
species.name="Bradypus",
ENSEMBLE.tune=TRUE,
ENSEMBLE.min = 0.65,
MAXENT=0, MAXNET=0, MAXLIKE=0, GBM=0, GBMSTEP=0, RF=1, CF=0,
GLM=1, GLMSTEP=0, GAM=0, GAMSTEP=0, MGCV=0, MGCVFIX=0,
DOMAIN=1, MAHAL=0, MAHAL01=0,
Yweights="BIOMOD",
factors="biome",
evaluations.keep=TRUE, models.keep=TRUE,
RF.formula=rfformula,
GLM.formula=glmformula)

# with option models.keep, all model objects are saved in ensemble object
# the same slots can be used to replace model objects with new calibrations
ensemble.nofactors$models$RF
summary(ensemble.nofactors$models$GLM)
ensemble.nofactors$models$BIOCLIM
ensemble.nofactors$models$DOMAIN
ensemble.nofactors$models$formulae

# evaluations are kept in different slot
attributes(ensemble.nofactors$evaluations)
plot(ensemble.nofactors$evaluations$RF.T, "ROC")

# 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.calibrate.models(x=predictors,
pt=pres_test, at=backg_test,
layer.drops="biome",
species.name="Bradypus",
ENSEMBLE.tune=TRUE,
ENSEMBLE.min=0.65,
SINK=TRUE,
MAXENT=0, MAXNET=0, MAXLIKE=0, GBM=0, GBMSTEP=0, RF=1, CF=0,
GLM=1, GLMSTEP=0, GAM=0,
GAMSTEP=0, MGCV=0, MGCVFIX=0, EARTH=0, RPART=0, NNET=0, FDA=0,
```r
ensemble.nofactors <- ensemble.calibrate.models(x=predictors,
p=pres_train, a=backg_train,
pt=pres_test, at=backg_test,
layer.drops="biome",
species.name="Bradypus",
SINK=TRUE,
ENSEMBLE.tune=TRUE,
ENSEMBLE.min=0.65,
MAXENT=0, MAXNET=0, MAXLIKE=0, GBM=0, GBMSTEP=0, RF=0, CF=0,
GLM=1, GLMSTEP=0, GAM=0, GAMSTEP=0, MGCV=0, MGCVFIX=0,
EARTH=0, RPART=0, NNET=0, FDA=0, SVM=0, SVME=0, GLMNET=0,
BIOLIM.O=0, BIOLIM=1, DOMAIN=1, MAHAL=0, MAHAL01=0,
Yweights="BIOMOD", evaluations.keep=TRUE,
formulae.defaults=TRUE)
```

# Instead of providing presence and background locations, provide data.frames.
# Because 'biome' is a factor, RasterStack needs 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.calibrate.models(x=predictors,
TrainData=TrainData1, TestData=TestData1,
p=pres_train, a=backg_train,
pt=pres_test, at=backg_test,
species.name="Bradypus",
SINK=TRUE,
ENSEMBLE.tune=TRUE,
ENSEMBLE.min=0.65,
MAXENT=0, MAXNET=1, MAXLIKE=1, GBM=1, GBMSTEP=0, RF=1, CF=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, GLMNET=1,
BIOLIM.O=1, BIOLIM=1, DOMAIN=1, MAHAL=0, MAHAL01=1,
Yweights="BIOMOD", factors="biome",
evaluations.keep=TRUE)
```

# Compare different methods of calculating ensembles
ensemble.factors2 <- ensemble.calibrate.models(x=predictors,
TrainData=TrainData1, TestData=TestData1,
p=pres_train, a=backg_train,
pt=pres_test, at=backg_test,
species.name="Bradypus",
Yweights="BIOMOD", factors="biome",
evaluations.keep=TRUE)
```

# test performance of different suitability models
# data are split in 4 subsets, each used once for evaluation
ensemble.nofactors2 <- ensemble.calibrate.weights(x=predictors,
p=pres, a=background, k=4,
species.name="Bradypus",
SINK=TRUE, PROBIT=TRUE,
MAXENT=0, MAXNET=1, MAXLIKE=1, GBM=1, GBMSTEP=0, RF=1, CF=1,
GLM=1, GLMSTEP=1, GAM=1, GAMSTEP=1, MGCV=1, MGCVFIX=1,
EARTH=1, RPART=1, NNET=1, SVM=1, SVME=1, GLMNET=1,
BIOLIM.O=1, BIOLIM=1, DOMAIN=1, MAHAL=0, MAHAL01=1,
ENSEMBLE.tune=TRUE,
ENSEMBLE.best=c(4:10), ENSEMBLE.exponent=c(1, 2, 3),
Yweights="BIOMOD", factors="biome",
evaluations.keep=TRUE)

ensemble.nofactors2$AUC.table
ensemble.nofactors2$eval.table.all

# 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,
p=pres, a=background, k=4,
species.name="Bradypus",
SINK=TRUE,
difference=TRUE,
VIF=TRUE, PROBIT=TRUE,
MAXENT=0, MAXNET=1, MAXLIKE=1, GBM=1, GBMSTEP=0, RF=1, CF=1,
GLM=1, GLMSTEP=1, GAM=1, GAMSTEP=1, MGCV=1, MGCVFIX=1,
EARTH=1, RPART=1, NNET=1, SVM=1, SVME=1, GLMNET=1,
BIOLIM.O=1, BIOLIM=1, DOMAIN=1, MAHAL=0, MAHAL01=1,
ENSEMBLE.tune=TRUE,
ENSEMBLE.best=0, ENSEMBLE.exponent=c(1, 2, 3),
ENSEMBLE.min=0.7,
Yweights="BIOMOD", factors="biome")

ensemble.variables

# use function ensemble.VIF to select a subset of variables
# factor variables are not handled well by the function
# and therefore factors are removed
# however, one can check for factors with car::vif through
# the ensemble.calibrate.models function
# VIF.analysis$var.drops can be used as input for ensemble.calibrate.models or
# ensemble.calibrate.weights

predictors <- stack(predictor.files)
predictors <- subset(predictors, subset=c("bio1", "bio5", "bio6", "bio8", "bio12", "bio16", "bio17", "biome"))

ensemble.pairs(predictors)

VIF.analysis <- ensemble.VIF(predictors, factors="biome")
VIF.analysis

# alternative solution where bio1 and bio12 are kept
VIF.analysis <- ensemble.VIF(predictors, factors="biome", keep=c("bio1", "bio12"))
VIF.analysis

## 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, new.levels = NULL, 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.
new.levels  character vector giving factor levels that are not encountered in xcat and for
which dummy layers should be created (this could be useful in dealing with
novel conditions)
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; ex-
pected 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 re-
placing 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.dummy.variables 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.calibrate.models, 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)
bio layer <- predictors[["biome"]]

# 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)
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
group <- kfold(pres, 5)
pres_train <- pres[group != 1, ]
pres_test <- pres[group == 1, ]

# choose background points
background <- randomPoints(predictors, n=1000, extf=1.00)
colnames(background)=c("lon", "lat")
groupa <- kfold(background, 5)
backg_train <- background[groupa != 1, ]
backg_test <- background[groupa == 1, ]

# note that dummy variables with no variation are not used by DOMAIN
# note that dummy variables are not used by MAHAL and MAHAL01
# (neither are categorical variables)
ensemble.nofactors <- ensemble.calibrate.models(x=predictors, p=pres_train, a=backg_train, pt=pres_test, at=backg_test,
species.name="Bradypus",
VIF=T,
MAXENT=1, MAXLIKE=1, GBM=1, GBMSTEP=0, RF=1, GLM=1, GLMSTEP=0, GAM=1,
GAMSTEP=0, MGCV=1, MGCVFIX=0, EARTH=1, RPART=1, NNET=1, FDA=1,
SVM=1, SVM=1, BIOCLIM.0=1, BIOCLIM=1, DOMAIN=1, MAHAL=0, MAHAL01=1,
Yweights="BIOMOD",
dummy.vars=c("biome_1", "biome_2", "biome_7", "biome_8", "biome_13"),
PLOTS=FALSE, evaluations.keep=TRUE)
### Description

Function `ensemble.novel` creates the map with novel conditions. Function `ensemble.novel.object` provides the reference values used by the prediction function used by `predict`.

### Usage

```r
ensemble.ecocrop(x = NULL, ecocrop.object = NULL, 
                  RASTER.object.name = ecocrop.object$name, RASTER.stack.name = x@title, 
                  RASTER.format = "raster", RASTER.datatype = "INT2S", RASTER.NAflag = -32767, 
                  KML.out = TRUE, KML.blur = 10, KML.maxpixels = 100000, 
                  CATCH.OFF = FALSE)

ensemble.ecocrop.object(temp.thresholds, rain.thresholds, name = "crop01", 
                         temp.multiply = 10, annual.temps = TRUE, transform = 1)
```

### Arguments

- **x**: RasterStack object (stack) containing all environmental layers for which suitability should be calculated.
- **ecocrop.object**: Object listing optimal and absolute minima and maxima for the rainfall and temperature values, used by the prediction function that is used internally by `predict`. This object is created with `ensemble.ecocrop.object`.
- **RASTER.object.name**: First part of the names of the raster file that will be generated, expected to identify the species or crop for which ranges were calculated.
- **RASTER.stack.name**: Last part of the names of the raster file that will be generated, expected to identify the predictor stack used.
- **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 TRUE, then kml files will be saved in a subfolder ‘kml/zones’.
- **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 $KML\cdot\text{blur}^2$, which may help avoid blurring of isolated pixels. See also KML.

CATCH.OFF Disable calls to function tryCatch.

temp.thresholds Optimal and absolute thresholds for temperatures. These will be sorted as: absolute minimum temperature, optimal minimum temperature, optimal maximum temperature and absolute maximum temperature.

rain.thresholds Optimal and absolute thresholds for annual rainfall. These will be sorted as: absolute minimum rainfall, optimal minimum rainfall, optimal maximum rainfall and absolute maximum rainfall.

name Name of the object, expect to expected to identify the species or crop

temp.multiply Multiplier for temperature values. Default of 10 is to be used with raster layers where temperature was multiplied by 10 such as Worldclim or AFRICLIM.

annual.temps If TRUE then temperature limits are assumed to apply to mean annual temperature (bioclimatic variable bio1). If FALSE then minimum temperature limits are assumed to apply to the temperature of the coldest month (bioclimatic variable bio6) and maximum temperature limits are assumed to apply to the temperature of the hottest month (bioclimatic variable bio5). See also biovars.

transform Exponent used to transform probability values obtained from interpolating between optimal and absolute limits. Exponent of 2 results in squaring probabilities, for example input probabilities of 0.5 transformed to $0.5^2 = 0.25$.

Details

Function ensemble.ecocrop maps suitability for a species or crop based on optimal and absolute temperature and rainfall limits. Where both temperature and rainfall are within the optimal limits, suitability of 1000 is calculated. Where both temperature and rainfall are outside the absolute limits, suitability of 0 is calculated. In situations where temperature or rainfall is in between the optimal and absolute limits, then suitability is interpolated between 0 and 1000, and the lowest suitability from temperature and rainfall is calculated. Setting very wide rainfall limits will simulate the effect of irrigation, i.e. where suitability only depends on temperature limits.

For a large range of crop and plant species, optimal and absolute limits are available from the FAO ecocrop database (http://www.fao.org/land-water/land/land-governance/land-resources-planning-toolbox/category/details/en/c/1027491/), hence the name of the function. A different implementation of suitability mapping based on ecocrop limits is available from ecocrop. Ecocrop thresholds for several species are available from: getCrop

Value

Function ensemble.ecocrop.object returns a list with following objects:

name name for the crop or species

temp.thresholds optimal and absolute minimum and maximum temperature limits

rain.thresholds optimal and absolute minimum and maximum annual rainfall limits
ensemble.ecocrop

annual.temps logical indicating whether temperature limits apply to annual temperatures
transform exponent to transform suitability values

Author(s)
Roeland Kindt (World Agroforestry Centre)

See Also
biovars

Examples

## Not run:
# test with Brazil nut (limits from FAO ecocrop)
# temperature: (12) 20-36 (40)
# annual rainfall: (1400) 2400-2800 (3500)

# 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", "bio12"))
predictors
predictors@title <- "base"

Brazil.ecocrop <- ensemble.ecocrop.object(temp.thresholds=c(20, 36, 12, 40),
   rain.thresholds=c(2400, 2800, 1400, 3500),
   annual.temps=FALSE, name="Bertholletia_excelsa")
Brazil.ecocrop
ensemble.ecocrop(predictors, ecocrop.object=Brazil.ecocrop)

dev.new()
par.old <- graphics::par(no.readonly=T)
graphics::par(mfrow=c(1,2))

rasterfull1 <- paste("ensembles//ecocrop//Bertholletia_excelsa_base.grd", sep="")
rasterfull1 <- raster(rasterfull1)
# raster file saved probabilities as integer values between 0 and 1000
rasterfull1 <- rasterfull1/1000
raster::plot(rasterfull1, main="Ecocrop suitability")

GBIFloc <- gbif(genus="Bertholletia", species="excelsa", geo=TRUE)
GBIFpres <- cbind(GBIFloc$lon, GBIFloc$lat)
GBIFpres <- GBIFpres[complete.cases(GBIFpres), ]
GBIFpres <- GBIFpres[duplicated(GBIFpres) == FALSE, ]
point.suitability <- extract(rasterfull1, y=GBIFpres)
point.suitability[is.na(point.suitability)] <- -1
```
GBIFpres.optimal <- GBIFpres[point.suitability == 1, ]
GBIFpres.suboptimal <- GBIFpres[point.suitability < 1 & point.suitability > 0, ]
GBIFpres.not <- GBIFpres[point.suitability == 0, ]

raster::plot(rasterfull1, main="GBIF locations",
             sub="blue: optimal, cyan: suboptimal, red: not suitable")
bg.legend <- c("blue", "cyan", "red")

points(GBIFpres.suboptimal, pch=21, cex=1.2, bg=bg.legend[2])
points(GBIFpres.optimal, pch=21, cex=1.2, bg=bg.legend[1])
points(GBIFpres.not, pch=21, cex=1.2, bg=bg.legend[3])

graphics::par(par.old)
```

## End(Not run)

---

### ensemble.evaluate

Model evaluation including True Skill Statistic (TSS), AUCDiff and Symmetric Extremal Dependence Index (SEDI).

#### Description

The main function of `ensemble.evaluate` calculates various model evaluation statistics. Function `ENSEMBLE.SEDI` calculates the Symmetric Extremal Dependence Index (SEDI) from the True Positive Rate (TPR = Sensitivity = Hit Rate) and the False Positive Rate (FPR = False Alarm Rate = 1 - Specificity).

#### Usage

```
ensemble.evaluate(eval, fixed.threshold = NULL, eval.train = NULL)
ensemble.SEDI(TPR, FPR, small = 1e-9)
```

#### Arguments

- **eval**: ModelEvaluation object (\texttt{evaluate}), ideally obtained via model testing data that were not used for calibrating the model.
- **fixed.threshold**: Absence-presence threshold to create the confusion matrix. See also \texttt{(threshold} and \texttt{ensemble.threshold}).
- **eval.train**: ModelEvaluation object (\texttt{evaluate}), ideally obtained via model calibration data that were used for calibrating the model.
- **TPR**: True Presence Rate, equal to correctly predicted presence observations divided by total number of presence observations. Also known as Sensitivity or Hit Rate.
False Presence Rate, equal to wrongly predicted absence observations divided by total number of absence observations. Also known as False Alarm Rate.

small amount that replaces zeroes in calculations.

**Details**

Details for the True Skill Statistic (TSS = TPR + TNR - 1 = TPR - FPR), Symmetric Extremal Dependence Index (SEDI), False Negative Rate (omission or miss rate) and AUCdiff (AUCtrain - AUCtest) are available from Ferro and Stephenson (2011), Wunderlich et al. (2019) and Castellanos et al. (2019).

Values for TSS and SEDI are given for the fixed absence-presence threshold, as well as their maximal values across the entire range of candidate threshold values calculate by `ensemble.evaluate`.

In case that `fixed.threshold` is not provided, it is calculated from the calibration ModelEvaluation as the threshold that maximizes the sum of TPR (sensitivity) and TNR (specificity) (and thus also maximizes the TSS for the calibration).

**Value**

A numeric vector with following values.

- **AUC**: Area Under The Curve for the testing ModelEvaluation
- **TSS**: maximum value of the True Skill Statistic over range of threshold values
- **SEDI**: maximum value of the Symmetric Extremal Dependence Index over range of threshold values
- **TSS.fixed**: True Skill Statistic at the fixed threshold value
- **SEDI.fixed**: SEDI at the fixed threshold value
- **FNR.fixed**: False Negative Rate (= omission rate) at the fixed threshold value
- **MCR.fixed**: Missclassification Rate at the fixed threshold value
- **AUCdiff**: Difference between AUC for calibration and the testing data

**Author(s)**

Roeland Kindt (World Agroforestry Centre)

**References**


**See Also**

`ensemble.batch`
Examples

```r
## check examples from Ferro and Stephenson (2011)
## see their Tables 2 - 5

TPR.Table2 <- 55/100
FPR.Table2 <- 45/900
ensemble.SEDI(TPR=TPR.Table2, FPR=FPR.Table2)

TPR.Table4 <- 195/300
FPR.Table4 <- 105/700
ensemble.SEDI(TPR=TPR.Table4, FPR=FPR.Table4)

## Not run:
## 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 <- "predictors"

# 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 (1/4) and training (3/4) data
group <- kfold(pres, 4)
pres_train <- pres[group != 1, ]
pres_test <- pres[group == 1, ]

# choose background points
background <- randomPoints(predictors, n=1000, extf=1.00)
colnames(background)=c('lon', 'lat')
groupa <- kfold(background, 4)
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)
# factors removed for BIOCLIM, DOMAIN, MAHAL
ensemble.nofactors <- ensemble.calibrate.models(x=predictors, p=pres_train, a=backg_train,
pt=pres_test, at=backg_test,
species.name="Bradypus",
ENSEMBLE.tune=TRUE,
ENSEMBLE.min = 0.65,
MAXENT=0, MAXNET=0, MAXLIKE=0, GBM=0, GBMSTEP=0, RF=1, CF=0,
GLM=1, GLMSTEP=0, GAM=0, GAMSTEP=0, MGCV=0, MGCVFIX=0,
EARTH=0, RPART=0, NNNE=0, FDA=0, SWM=0, SVM=0, GLMNET=0,
BIOCLIM.O=0, BIOCLIM=1, DOMAIN=1, MAHAL=0, MAHAL01=0,
Yweights="BIOMOD",
factors="biome",
evaluations.keep=TRUE, models.keep=FALSE,
RF.formula=rfformula,
GLM.formula=glmformula)

# with option evaluations.keep, all model evaluations are saved in the ensemble object
attributes(ensemble.nofactors$evaluations)

# Get evaluation statistics for the ENSEMBLE model
eval.ENSEMBLE <- ensemble.nofactors$evaluations$ENSEMBLE.T
eval.calibrate.ENSEMBLE <- ensemble.nofactors$evaluations$ENSEMBLE.C
ensemble.evaluate(eval=eval.ENSEMBLE, eval.train=eval.calibrate.ENSEMBLE)

# TSS is maximum where specificity + sensitivity is maximum
threshold.specsens <- threshold(eval.ENSEMBLE, stat="spec_sens")
ensemble.evaluate(eval=eval.ENSEMBLE, fixed.threshold=threshold.specsens,
                   eval.train=eval.calibrate.ENSEMBLE)

# usual practice to calculate threshold from calibration data
ensemble.evaluate(eval=eval.ENSEMBLE, eval.train=eval.calibrate.ENSEMBLE)

## End(Not run)

---

**Description**

Function `ensemble.novel` creates the map with novel conditions. Function `ensemble.novel.object` provides the reference values used by the prediction function used by `predict`. 

---

*ensemble.novel*  Mapping of novel environmental conditions (areas where some of the environmental conditions are outside the range of environmental conditions of a reference area).
Usage

ensemble.novel(x = NULL, novel.object = NULL,
  RASTER.object.name = novel.object$name, RASTER.stack.name = x@title,
  RASTER.format = "raster", RASTER.datatype = "INT1S", RASTER.NAflag = -127,
  KML.out = FALSE, KML.maxpixels = 100000, KML.blur = 10,
  CATCH.OFF = FALSE)

ensemble.novel.object(x = NULL, name = "reference1", mask.raster = NULL,
  quantiles = FALSE, probs = c(0.05, 0.95), factors = NULL)

Arguments

x RasterStack object (stack) containing all environmental layers for which novel
  conditions should be calculated. With ensemble.novel.object, x can also be
  a data.frame.
novel.object Object listing minima and maxima for the environmental layers, used by the
  prediction function that is used internally by predict. This object is created
  with ensemble.novel.object.
RASTER.object.name First part of the names of the raster file that will be generated, expected to iden-
  tify the area and time period for which ranges were calculated
RASTER.stack.name Last part of the names of the raster file that will be generated, expected to iden-
  tify the predictor stack used
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 TRUE, then kml files will be saved in a subfolder 'kml/zones'.
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.
CATCH.OFF Disable calls to function tryCatch.
name Name of the object, expect to expected to identify the area and time period for
  which ranges were calculated and where no novel conditions will be detected
mask.raster RasterLayer object (raster) that can be used to select the area for which reference
  values are obtained (see mask)
quantiles If TRUE, then replace minima and maxima with quantile values. See also quantile
  and quantile
probs Numeric vector of probabilities [0, 1] as used by quantile and quantile
factors vector that indicates which variables are factors; these variables will be ignored
  for novel conditions
Details

Function `ensemble.novel` maps zones (coded '1') that are novel (outside the minimum-maximum range) relative to the range provided by function `ensemble.novel.object`. Values that are not novel (inside the range of minimum-maximum values) are coded '0'. In theory, the maps show the same areas that have negative Multivariate Environmental Similarity Surface (MESS) values ((mess))

Value

Function `ensemble.novel.object` returns a list with following objects:

- **minima**: minima of the reference environmental conditions
- **maxima**: maxima of the reference environmental conditions
- **name**: name for the reference area and time period

Author(s)

Roeland Kindt (World Agroforestry Centre)

See Also

`ensemble.raster`, `ensemble.bioclim` and `ensemble.bioclim.graph`

Examples

```r
# 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)
predictors <- subset(predictors, subset=c("bio1", "bio5", "bio6", "bio7", "bio8", "bio12", "bio16", "bio17"))
predictors
predictors$titles <- "base"

# reference area to calculate environmental ranges
ext <- extent(-70, -50, -10, 10)
extent.values2 <- c(-70, -50, -10, 10)
predictors.current <- crop(predictors, y=ext)
predictors.current <- stack(predictors.current)
	novel.test <- ensemble.novel.object(predictors.current, name="noveltest")
novel.test

novel.raster <- ensemble.novel(x=predictors, novel.object=novel.test, KML.out=T)
novel.raster

plot(novel.raster)
# no novel conditions within reference area
rect(extent.values2[1], extent.values2[3], extent.values2[2], extent.values2[4])
```
# use novel conditions as a simple species suitability mapping method
# presence points
presence_file <- paste(system.file(package="dismo"), 
"/ex/bradypus.csv", sep='')
pres <- read.table(presence_file, header=TRUE, sep=';', [,,-1]
pres.data <- data.frame(extract(predictors, y=pres))

# ranges and maps
Bradypus.ranges1 <- ensemble.novel.object(pres.data, name="Bradypus", quantiles=F)
Bradypus.ranges1
Bradypus.novel1 <- ensemble.novel(x=predictors, novel.object=Bradypus.ranges1, KML.out=T)
Bradypus.novel1

par.old <- graphics::par(no.readonly=T)
graphics::par(mfrow=c(1,2))

# suitable where there are no novel conditions
raster::plot(Bradypus.novel1, breaks=c(-0.1, 0, 1), col=c("green", "grey"),
main="Suitability mapping using minimum to maximum range")
points(pres[, 2] - pres[, 1], pch=1, col="red", cex=0.8)

# use 90 percent intervals similar to BIOCLIM methodology
Bradypus.ranges2 <- ensemble.novel.object(pres.data, name="BradypusQuantiles", quantiles=T)
Bradypus.ranges2
Bradypus.novel2 <- ensemble.novel(x=predictors, novel.object=Bradypus.ranges2, KML.out=T)
Bradypus.novel2
raster::plot(Bradypus.novel2, breaks=c(-0.1, 0, 1), col=c("green", "grey"),
main="Suitability mapping using quantile range")
points(pres[, 2] - pres[, 1], pch=1, col="red", cex=0.8)

graphics::par(par.old)

# deal with novel factor levels through dummy variables
predictors <- stack(predictor.files)
bio.layer <- predictors["biome"]
bio.layer
ensemble.dummy.variables(xcat=bio.layer, most.frequent=0, freq.min=1,
overwrite=TRUE)
predictors.dummy <- stack(predictor.files)
predictors.dummy <- subset(predictors.dummy, subset=c("biome_1", "biome_2", "biome_3",
"biome_4", "biome_5", "biome_7", "biome_8", "biome_9",
"biome_10", "biome_12", "biome_13", "biome_14"))
predictors.dummy
predictors.dummy@title <- "base_dummy"
predictors.dummy.current <- crop(predictors.dummy, y=ext)
predictors.dummy.current <- stack(predictors.dummy.current)

novel.levels <- ensemble.novel.object(predictors.dummy.current, name="novellevels")
novel.levels
novel.levels.raster <- ensemble.novel(x=predictors.dummy, novel.object=novel.levels,
KML.out=T)
```r
novel.levels.raster

novel.levels.quantiles <- ensemble.novel.object(predictors.dummy.current, quantiles=TRUE, name="novellevels_quantiles")
novel.levels.quantiles

novel.levels.quantiles.raster <- ensemble.novel(x=predictors.dummy, novel.object=novel.levels.quantiles, KML.out=T)
novel.levels.quantiles.raster

# difference in ranges for variables with low frequencies
background <- dismo::randomPoints(predictors.dummy.current, n=10000, p=NULL, excludep=F)
eextract.data <- extract(predictors.dummy.current, y=background)
colSums(extract.data)/sum(extract.data)*100

novel.levels

par.old <- graphics::par(no.readonly=T)

graphics::par(mfrow=c(1,2))
raster::plot(novel.levels.raster, breaks=c(-0.1, 0, 1), col=c("grey", "green"),
            main="novel outside minimum to maximum range")
rect(extent.values2[1], extent.values2[3], extent.values2[2], extent.values2[4])
raster::plot(novel.levels.quantiles.raster, breaks=c(-0.1, 0, 1), col=c("grey", "green"),
            main="novel outside quantile range")
rect(extent.values2[1], extent.values2[3], extent.values2[2], extent.values2[4])

graphics::par(par.old)

## End(Not run)
```

---

**ensemble.PET.season**  
*Calculate the balance between precipitation and potential evapotranspiration for the dry season with the largest balance (maximum climatological water deficit, accumulated aridity).*

---

**Description**

Internally, the function first determines different dry seasons, defined by consecutive months where precipitation is smaller than potential evapotranspiration. The function then returns the summation of monthly balances of precipitation minus potential evapotranspiration that is largest (most negative) of the different dry seasons.

**Usage**

```r
ensemble.PET.season(PREC.stack = NULL, PET.stack = NULL, 
                   filename = NULL, overwrite = TRUE, 
                   CATCH.OFF = FALSE, ...)
```
ensemble.PET.season

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREC.stack</td>
<td>stack object (stack) with monthly precipitation values.</td>
</tr>
<tr>
<td>PET.stack</td>
<td>stack object (stack) with monthly potential evapotranspiration values.</td>
</tr>
<tr>
<td>filename</td>
<td>Name for writing the resulting raster layer (as in writeRaster).</td>
</tr>
<tr>
<td>overwrite</td>
<td>Replace a previous version of the same file.</td>
</tr>
<tr>
<td>CATCH.OFF</td>
<td>Disable calls to function tryCatch.</td>
</tr>
<tr>
<td>...</td>
<td>Additional arguments for writeRaster.</td>
</tr>
</tbody>
</table>

Details

Unlike the methodology described by Chave et al. (2014), the assumption is not made that there is a single drought season. Internally, the function first identifies dry months as months where the balance of precipitation minus potential evapotranspiration is negative. Then dry seasons are identified as consecutive dry months. For each dry season, the total sum of balances is calculated. The function finally identifies and returns the largest of these balances.

The algorithm of the function should obtain the same values of the Maximum Cumulative Water Deficit as from rules described by Aragao et al. 2007 (section 2.2), when using fixed monthly PET values of 100 mm instead of calculated monthly PET values (calculated, for example, from monthly mean temperatures and extraterrestrial solar radiation through the Hargreaves method).

Note that calculation may take a while for larger raster data sets.

Value

The function returns and writes a raster layer

Author(s)

Roeland Kindt (World Agroforestry Centre)

References


See Also

ensemble.batch

Examples

```r
## Not run:
```
library(raster)
stack1 <- stack(monthly.prec.files)
stack2 <- stack(monthly.PET.files)
# note that the stacks should be of the same extend and resolution
ENSEMBLE.PET.season(PREC.stack=stack1, PET.stack=stack2,
    filename=paste(getwd(), 'Aridity.deficit.tif', sep=''))

## End(Not run)

Description

The main function of `ensemble.PET.seasons` calculates the number of growing seasons and their starts and lengths from the dry period criterion of $2 \times P < PET$ [http://www.fao.org/nr/climpag/cropfor/lgp_en.asp](http://www.fao.org/nr/climpag/cropfor/lgp_en.asp). Functions `ensemble.PREC.season` and `ensemble.TMEAN.season` calculate the total precipitation and average temperature for provided starts and lengths of a growing season. Together with data on optimal and absolute precipitation and temperature limits for a selected crop (as available from FAO’s ECOCROP database), these layers enable the calculation of crop suitability using methods detailed in Chapman et al. (2020).

Usage

```r
ensemble.PET.seasons(PREC.stack=NULL, PET.stack=NULL,
    index=c("seasons", "start1", "length1", "start2", "length2", "start3", "length3"),
    filename=NULL, overwrite=TRUE,
    CATCH.OFF=FALSE, ...)

ensemble.prec.season(PREC.stack=NULL,
    start.layer=NULL, length.layer=NULL,
    filename=NULL, overwrite=TRUE,
    CATCH.OFF=FALSE, ...)

ensemble.tmean.season(TMEAN.stack=NULL,
    start.layer=NULL, length.layer=NULL,
    filename=NULL, overwrite=TRUE,
    CATCH.OFF=FALSE, ...)

ensemble.season.suitability(season.raster=NULL,
    thresholds=NULL,
    filename=NULL, overwrite=TRUE,
    CATCH.OFF=FALSE, ...)
```
Arguments

- **PREC.stack**: stack object (**stack**) with monthly precipitation values.
- **PET.stack**: stack object (**stack**) with monthly potential evapotranspiration values.
- **TMEAN.stack**: stack object (**stack**) with monthly average temperature values.
- **index**: selection of type of output - see details.
- **start.layer**: raster layer with index of the month of the start of the growing season.
- **length.layer**: raster layer with index of the length of the growing season.
- **season.raster**: raster layer with seasonal precipitation or mean temperature.
- **thresholds**: optimal and absolute thresholds of crop suitability, defined similarly as by ECOCROP.
- **filename**: Name for writing the resulting raster layer (as in `writeRaster`).
- **overwrite**: Replace a previous version of the same file.
- **CATCH.OFF**: Disable calls to function **tryCatch**.
- **...**: Additional arguments for `writeRaster`.

Details

Function `ensemble.PET.seasons` calculates the number, starts and lengths of growing seasons after first internally determining dry periods from the criterion of $2 \times P < PET$. The function was developed with data sets with monthly precipitation and PET values, but probably can also work with data sets of other temporal resolution. Where there are multiple gaps between dry seasons, different growing periods are identified.

The definition of dry periods is less strict than the definition of $P < PET$ used in `ensemble.PET.season`, following the methodologies for this function.

Argument **index** determines the contents of the output rasters: - **seasons** selects the number of growing periods to be returned; - **start1** selects the index of the start of the first or only growing period to be returned; - **length1** selects the index of the end of the first or only growing period to be returned; - **start2** selects the index of the start of the second growing period to be returned; - **length2** selects the index of the end of the second growing period to be returned; - **start3** selects the index of the start of the third growing period to be returned; and - **length3** selects the index of the end of the third growing period to be returned.

The methodology of calculating crop suitability is directly based on Chapman et al. (2020), following their equations 2 (temperature suitability, based on the mean temperature of the growing season) and 3 (precipitation suitability, based on the total precipitation of the growing season). The methods of Chapman et al. (2020) are based on Ramirez-Villegas et al. (2013), including the calculation of crop suitability as the product of temperature suitability and crop suitability (their respective equations 1 and 3).

Crop thresholds are available from the FAO ECOCROP database, which are also available via function **getCrop**.

Note that calculations can take a while for larger data sets.

Value

The function returns and writes raster layers.
ensemble.PET.seasons

Author(s)
Roeland Kindt (World Agroforestry Centre)

References


See Also
ensemble.PET.season

Examples

```r
## Not run:
library(raster)
P.stack <- stack(monthly.prec.files)
PET.stack <- stack(monthly.PET.files)

# Calculate average monthly values similarly as in
TMIN.stack <- stack(monthly.tmin.files)
TMAX.stack <- stack(monthly.tmax.files)
T.stack <- stack(0.5*(TMIN.stack + TMAX.stack))

# step 1: determine number of seasons, start and length of season 1
seasons.raster <- ensemble.PET.seasons(PREC.stack=P.stack, PET.stack=PE.stack,
                                        index="seasons", filename="seasons.tif", CATCH.OFF=TRUE)
start1.raster <- ensemble.PET.seasons(PREC.stack=P.stack, PET.stack=PE.stack,
                                       index="start1", filename="start1.tif", CATCH.OFF=TRUE)
length1.raster <- ensemble.PET.seasons(PREC.stack=P.stack, PET.stack=PE.stack,
                                       index="length1", filename="length1.tif", CATCH.OFF=TRUE)
start2.raster <- ensemble.PET.seasons(PREC.stack=P.stack, PET.stack=PE.stack,
                                       index="start2", filename="start2.tif", CATCH.OFF=TRUE)
length2.raster <- ensemble.PET.seasons(PREC.stack=P.stack, PET.stack=PE.stack,
                                       index="length2", filename="length2.tif", CATCH.OFF=TRUE)

# step 2: calculate total precipitation in first rainy season,
# then use this value to calculate precipitation suitability
```

## Not run:
ensemble.raster <- ensemble.prec.season(PREC.stack=P.stack, 
start.layer=start1.raster, length.layer=length1.raster, 
filename="precSeason.tif", CATCH.OFF=FALSE)

dismo::getCrop("Sorghum (med. altitude)")

prec.suit <- ensemble.season.suitability(season.raster=prec.season, 
thresholds=c(300, 500, 1000, 3000), 
filename="precSuitability.tif", CATCH.OFF=FALSE)

# step 3: calculate average temperature in first rainy season, 
# then use this value to calculate temperature suitability

tmean.season <- ensemble.tmean.season(TMEAN.stack=T.stack, 
start.layer=start1.raster, length.layer=length1.raster, 
filename="tmeanSeason.tif", CATCH.OFF=FALSE)

temp.suit <- ensemble.season.suitability(season.raster=tmean.season, 
thresholds=c(10, 24, 35, 40), 
filename="tempSuitability.tif", CATCH.OFF=FALSE)

# step 4: seasonal crop suitability is product of precipitation suitability 
# and temperature suitability

sorghum.suit <- prec.suit * temp.suit 
plot(sorghum.suit)

## End(Not run)

**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 Mahalonobis 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.
### ensemble.raster

**Usage**

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

```r
ensemble.habitat.change(base.map = file.choose(), 
    other.maps = utils::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)
```

```r
ensemble.area(x=NULL, km2=TRUE)
```

**Arguments**

- **xn**
  RasterStack object (stack) containing all layers that correspond to explanatory variables of an ensemble calibrated earlier with `ensemble.calibrate.models`. See also `predict`.

- **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.calibrate.weights` can be used.

- **thresholds**
  array with the threshold values separating predicted presence for each of the algorithms.

- **RASTER.species.name**
  First part of the names of the raster files that will be generated, expected to identify the modelled species (or organism).

- **RASTER.stack.name**
  Last part of the names of the raster files that will be generated, expected to identify the predictor stack used.

- **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`.
ensemble.raster

RASTER.models.overwrite
Overwrite the raster files that correspond to each suitability modelling algorithm
(if TRUE). (Overwriting actually implies that raster files are created or overwrit-
ten that start with "working_.").

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.

evaluate
if TRUE, then evaluate the created raster layers at locations p, a, pt and at (if
provided). See also evaluate

SINK
Append the results to a text file in subfolder 'outputs' (if TRUE). The name of
file is based on argument RASTER.species.name. In case the file already exists,
then results are appended. See also sink.

p
presence points used for calibrating the suitability models, typically available in
2-column (x, y) or (lon, lat) dataframe; see also prepareData and extract

a
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, typicall available
in 2-column (lon, lat) dataframe; see also prepareData and extract

catch.OFF
Disable calls to function tryCatch.

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.

x
RasterLayer object (raster) in a longitude-latitude coordinate system

km2
Provide results in square km rather than square m. See also areaPolygon

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 individ-
ual 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.calibrate.models`. The main difference between the two functions is that `ensemble.raster` generates raster layers for individual suitability models, whereas the purpose of `ensemble.calibrate.models` 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 - `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).

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]).

The `ensemble.area` function calculates the area of different categories with `areaPolygon`.

**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


See Also

evaluation.strip.plot, ensemble.calibrate.models, ensemble.calibrate.weights, 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"))
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
background <- randomPoints(predictors, n=1000, 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: determine weights through 4-fold cross-validation
ensemble.calibrate.step1 <- ensemble.calibrate.weights(
  x=predictors, p=pres, a=background, k=4,
  SINK=TRUE, species.name="Bradypus",
  MAXENT=0, MAXNET=1, MAXLIKE=1, GBM=1, GBMSTEP=0, RF=1, CF=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, GLMNET=1, BIOCLIM.O=1, BIOCLIM=1, DOMAIN=1, MAHAL=0, MAHAL01=1, ENSEMBLE.tune=TRUE, PROBIT=TRUE, ENSEMBLE.best=0, ENSEMBLE.exponent=c(1, 2, 3), ENSEMBLE.min=c(0.65, 0.7), Yweights="BIOMOD", PLOTS=FALSE, formulae.defaults=TRUE)

# step 1 generated the weights for each algorithm
model.weights <- ensemble.calibrate.step1$output.weights
x.batch <- ensemble.calibrate.step1$x
p.batch <- ensemble.calibrate.step1$p
a.batch <- ensemble.calibrate.step1$a
MAXENT.a.batch <- ensemble.calibrate.step1$MAXENT.a
factors.batch <- ensemble.calibrate.step1$factors
dummy.vars.batch <- ensemble.calibrate.step1$dummy.vars

# step 2: calibrate models with all available presence locations
# weights determined in step 1 calculate ensemble in step 2
ensemble.calibrate.step2 <- ensemble.calibrate.models(
  x=x.batch, p=p.batch, a=a.batch, MAXENT.a=MAXENT.a.batch,
  factors=factors.batch, dummy.vars=dummy.vars.batch,
  SINK=TRUE, species.name="Bradypus",
  models.keep=TRUE,
  input.weights=model.weights,
  ENSEMBLE.tune=FALSE, PROBIT=TRUE,
  Yweights="BIOMOD",
  PLOTS=FALSE, formulae.defaults=TRUE)

# step 3: use previously calibrated models to create ensemble raster layers
# re-evaluate the created maps at presence and background locations
# (note that re-evaluation will be different due to truncation of raster layers
# as they were saved as integer values ranging 0 to 1000)
ensemble.raster.results <- ensemble.raster(xn=predictors,
  models.list=ensemble.calibrate.step2$models,
  input.weights=model.weights,
  SINK=TRUE, evaluate=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.raster.results)
# are contrasted with a newly calibrated model (ensemble.raster.results2)
# step 1: 4-fold cross-validation
ensemble.calibrate2.step1 <- ensemble.calibrate.weights(
  x=x.batch, p=p.batch, a=a.batch, MAXENT.a=MAXENT.a.batch,
  factors=factors.batch, dummy.vars=dummy.vars.batch,
  k=4,
  SINK=TRUE, species.name="Bradypus",
  MAXENT=0, MAXNET=1, MAXLIKE=1, GBM=1, GBMSTEP=0, RF=1, CF=1,
  GLM=1, GLMSTEP=1, GAM=1, GAMSTEP=1, MGCV=1, MGCVFIX=1,
ensemble.raster

EARTH=1, RPART=1, NNET=1, FDA=1, SVM=1, SVME=1, GLMNET=1,
BIOCLIM.O=1, BIOCLIM=1, DOMAIN=1, MAHAL=0, MAHAL01=1,
ENSEMBLE.tune=TRUE, PROBIT=TRUE,
ENSEMBLE.best=0, ENSEMBLE.exponent=c(1, 2, 3),
ENSEMBLE.min=c(0.65, 0.7),
Yweights="BIOMOD",
PLOTS=FALSE, formulae.defaults=TRUE)

model.weights2 <- ensemble.calibrate2.step1$output.weights

ensemble.calibrate2.step2 <- ensemble.calibrate.models(
x=x.batch, p=p.batch, a=a.batch, MAXENT.a=MAXENT.a.batch,
factors=factors.batch, dummy.vars=dummy.vars.batch, SINK=TRUE, species.name="Bradypus",
models.keep=TRUE, input.weights=model.weights2,
ENSEMBLE.tune=FALSE, PROBIT=TRUE, Yweights="BIOMOD",
PLOTS=FALSE, formulae.defaults=TRUE)

ensemble.raster.results2 <- ensemble.raster(
xn=predictors, models.list=ensemble.calibrate2.step2$models,
input.weights=model.weights2, SINK=TRUE, evaluate=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")

change.file <- paste(getwd(), "/ensembles/change/Bradypus_recalibrated_presence.grd", sep="")

par.old <- graphics::par(no.readonly=T)
dev.new()
par(mfrow=c(2,2))
raster::plot(raster(base.file), breaks=c(-1, 0, 1), col=c("grey", "green"),
legend.shrink=0.8, main="base presence")
raster::plot(raster(other.file), breaks=c(-1, 0, 1), col=c("grey", "green"),
legend.shrink=0.8, main="other presence")
raster::plot(raster(change.file), breaks=c(-1, 0, 1, 10, 11),
col=c("grey", "blue", "red", "green"),
legend.shrink=0.8, main="habitat change", sub="11 remaining, 10 lost, 1 new")
graphics::par(par.old)

areas <- ensemble.area(raster(change.file))
areas

## End(Not run)
**ensemble.red**

Area of Occupancy (AOO) and Extent of Occurrence (EOO) via the `red` library.

---

**Description**

Function `ensemble.red` is a wrapper function for estimation of AOO and EOO computed for redlisting of species based on IUCN criteria (https://www.iucnredlist.org/about/regional).

Function `ensemble.chull.create` creates a mask layer based on a convex hull around known presence locations, inspired by `mcp` argument of the `map.sdm` function.

**Usage**

```r
ensemble.red(x)

ensemble.chull.create(x.pres = NULL, p = NULL, buffer.width = 0.2,
                      buffer.maxmins = FALSE, lonlat.dist = FALSE,
                      poly.only = FALSE,
                      RASTER.format = "raster", RASTER.datatype = "INT1U", RASTER.NAflag = 255,
                      overwrite = TRUE, ...)

ensemble.chull.apply(x.spec = NULL, mask.layer=NULL, keep.old=T,
                      RASTER.format="raster", RASTER.datatype="INT1U", RASTER.NAflag=255,
                      overwrite=TRUE, ...)

ensemble.chull.buffer.distances(p = NULL,
                                buffer.maxmins = FALSE, lonlat.dist = FALSE)

ensemble.chull.MSDM(p = NULL, a = NULL, species.name = NULL,
                    suit.file = NULL, suit.divide = 1000, MSDM.dir = NULL,
                    method = "BMCP", threshold = "spec_sens",
                    buffer = "species_specific")
```

**Arguments**

- **x**  
  RasterLayer object (`raster`), representing 'count' suitability layers (available from the 'count' and 'consensuscount' subdirectories of the 'ensembles' directory)

- **x.pres**  
  RasterLayer object (`raster`), representing 'presence' suitability layers (available from the 'presence' and 'consensuspresence' subdirectories of the 'ensembles' directory)

- **p**  
  known presence locations, available in 2-column (lon, lat) dataframe; see also `prepareData` and `extract`
buffer.width multiplier to create buffer (via gBuffer) by multiplying the maximum distance among the presence locations (calculated via pointDistance).

buffer.maxmins Calculate the buffer width based on the two neighbouring locations that are furthest apart (maximum of minimum distances from each location).

lonlat.dist Estimate the distance in km for longitude latitude data.

dataFormat Only return the polygon with the convex hull, but do not create the mask layer.

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.

overwrite Overwrite existing raster files. See writeRaster.

... Additional arguments for writeRaster.

x.spec RasterLayer object (raster), representing any suitability layer for the species under investigation.

mask.layer RasterLayer object (raster), representing the mask based on the convex hull around known presence locations. The function will replace all values in x.spec to zero where corresponding values in the mask.layer are zero.

keep.old keep a copy of the RasterLayer before the mask is applied.

absence of background locations, available in 2-column (lon, lat) dataframe.

species.name name of the species, ideally without spaces.

suit.file file with raster data corresponding to suitability values of the focal species.

suit.divide number by which values in the suitability raster should be divided to result in probabilities (BiodiversityR saves data as 1000 * suitability, hence these values need to be divided by 1000).

MSDM.dir name of the directory where input and processed raster files will be saved.

method method for MSDM_Posteriori function from c("OBR", "PRES", "LQ", "MCP", "BMCP").

threshold threshold for MSDM_Posteriori function from c("kappa", "spec_sens", "no_omission", "prevalence", "equal_sens_spec", "sensitivity").

buffer buffer for MSDM_Posteriori function.

Details

Function ensemble.red calculates AOO (aoo) and EOO (aoo) statistics calculated for areas with different consensus levels on species presence (1 model predicting presence, 2 models predicting presence, ...). In case that these statistics are within IUCN criteria for Critically Endangered (CR), Endangered (EN) or Vulnerable (VU), then this information is added in columns documenting the types of AOO and EOO.

Function ensemble.chull.create first creates a convex hull around known presence locations. Next, a buffer is created around the convex hull where the width of this buffer is calculated as the maximum distance among presence locations (pointDistance) multiplied by argument buffer.width. Finally, the mask is created by including all polygons of predicted species presence that are partially covered by the convex hull and its buffer.
Value

Function `ensemble.red` returns an array with AOO and EOO. Function `ensemble.chull.create` creates a mask layer based on a convex hull around known presence locations. Function `ensemble.chull.MSDM` prepares the input data and script for the MSDM_Posteriori function of the MSDM package.

Author(s)

Roeland Kindt (World Agroforestry Centre)

References

Cardoso P. 2017. red - an R package to facilitate species red list assessments according to the IUCN criteria. Biodiversity Data Journal 5:e20530. doi: 10.3897/BDJ.5.e20530


See Also

`ensemble.batch`

Examples

```r
## Not run:
## 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"))
predictors
predictors@title <- "red"

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

# fit 5 ensemble models (could take some time!)
# (examples for the red package use 100 models)
ensembles <- ensemble.batch(x=predictors,
                           xn=c(predictors),
```
```r
species.presence=pres,
thin.km=100,
k.splits=4, k.test=0,
n.ensembles=5,
SINK=TRUE,
ENSEMBLE.best=10, ENSEMBLE.exponent=c(1, 2, 3),
ENSEMBLE.min=0.6,
MAXENT=0, MAXNET=1, MAXLIKE=1, GBM=1, GBMSTEP=0, RF=1, CF=1,
GLM=1, GLMSTEP=1, GAM=1, GAMSTEP=1, MGCV=1, MGCVFIX=1,
EARTH=1, RPART=1, NN=1, FDA=1, SWM=1, SWME=1,
BIOLIM.O=1, BIOLIM=1, DOMAIN=1, MAHAL=0, MAHAL01=1,
PROBIT=TRUE,
Yweights="BIOMOD",
formulae.defaults=TRUE)

# first application of ensemble.red before applying the convex hull mask
# AOO and EOO are determined for each count level
library(red)
count.file <- paste(getwd(),
    "\ensembles/consensuscount/Bradypus variegatus_red.grd", sep="")
count.raster <- raster(count.file)
ensemble.red(count.raster)

# do not predict presence in polygons completely outside convex hull
# of known presence locations
pres.file <- paste(getwd(),
    "\ensembles/consensuspresence/Bradypus variegatus_red.grd", sep="")
pres.raster <- raster(pres.file)
pres <- pres[, -1]
chull.created <- ensemble.chull.create(x.pres=pres.raster, p=pres)

mask.raster <- chull.created$mask.layer
mask.poly <- chull.created$convex.hull

pres.chull <- ensemble.chull.apply(pres.raster, mask=mask.raster, keep.old=T)

# load previous for plotting
pres.file.old <- paste(getwd(),
    "\ensembles/consensuspresence/Bradypus variegatus_red_old.grd", sep="")
pres.raster.old <- raster(pres.file.old)

par.old <- graphics::par(no.readonly=T)
par(mfrow=c(1,2))
plot(pres.raster.old, breaks=c(-1, 0, 1), col=c("grey", "green"),
    main="before convex hull")
points(pres[, -1], col="blue")

# load new
pres.file <- paste(getwd(),
    "\ensembles/consensuspresence/Bradypus variegatus_red.grd", sep="")
pres.raster <- raster(pres.file)
plot(pres.raster, breaks=c(-1, 0, 1), col=c("grey", "green"),
    main="after convex hull")
```

plot(mask.poly, add=T, border="blue")

# new application of ensemble.red
dev.new()
plot(count.raster, main="before convex hull")
ensemble.red(count.raster)

# all cells where species is predicted not to be present according to the mask layer
# will be modified to a count of zero
count.chull <- ensemble.chull.apply(count.raster, mask=mask.raster, keep.old=T)

# load new
count.file <- paste(getwd(),
    "/ensembles/consensuscount/Bradypus variegatus_red.grd", sep="")
count.raster <- raster(count.file)
dev.new()
plot(count.raster, main="after convex hull")

# par.old <- graphics::par(no.readonly=T)
# create a smaller hull (0.05 * largest distance)
# First write back the original absence-presence file
pres.file.original <- paste(getwd(),
    "/ensembles/consensuspresence/Bradypus variegatus_red_old.grd", sep="")
pres.file <- paste(getwd(),
    "/ensembles/consensuspresence/Bradypus variegatus_red.grd", sep="")
pres.raster <- raster(pres.file.original)

# save as the original file
writeRaster(pres.raster, filename=pres.file, overwrite=T)
pres.raster <- raster(pres.file)

chull.created <- ensemble.chull.create(x.pres=pres.raster, p=pres1,
    buffer.width=0.05, lonlat.dist=TRUE)
mask.raster <- chull.created$mask.layer
mask.poly <- chull.created$convex.hull
pres.chull <- ensemble.chull.apply(pres.raster, mask=mask.raster, keep.old=T)

# load previous for plotting
pres.file.old <- paste(getwd(),
    "/ensembles/consensuspresence/Bradypus variegatus_red_old.grd", sep="")
pres.raster.old <- raster(pres.file.old)

par(mfrow=c(1,2))
plot(pres.raster.old, breaks=c(-1, 0, 1), col=c("grey", "green"),
    main="before convex hull")
points(pres1, col="blue")

# load new
pres.file <- paste(getwd(),
    "/ensembles/consensuspresence/Bradypus variegatus_red.grd", sep="")
pres.raster <- raster(pres.file)

plot(pres.raster, breaks=c(-1, 0, 1), col=c("grey", "green"),
    main="after convex hull")
plot(mask.poly, add=T, border="blue")
# create a hull based on the distance to the location with the farthest neighbour
# First write back the original absence-presence file
pres.file.original <- paste(getwd(),
   "/ensembles/consensuspresence/Bradypus variegatus_red_old.grd", sep="")
pres.file <- paste(getwd(),
   "/ensembles/consensuspresence/Bradypus variegatus_red.grd", sep="")
pres.raster <- raster(pres.file.original)
# save as the original file
writeRaster(pres.raster, filename=pres.file, overwrite=T)
pres.raster <- raster(pres.file)
chull.created <- ensemble.chull.create(x.pres=pres.raster, p=pres1,
   buffer.maxmins=TRUE, buffer.width=0.9, lonlat.dist=TRUE)
mask.raster <- chull.created$mask.layer
mask.poly <- chull.created$convex.hull
pres.chull <- ensemble.chull.apply(pres.raster, mask=mask.raster, keep.old=T)

# load previous for plotting
pres.file.old <- paste(getwd(),
   "/ensembles/consensuspresence/Bradypus variegatus_red_old.grd", sep="")
pres.raster.old <- raster(pres.file.old)
par(mfrow=c(1,2))
plot(pres.raster.old, breaks=c(-1, 0, 1), col=c("grey", "green"),
   main="before convex hull")
points(pres1, col="blue")

# load new
pres.file <- paste(getwd(),
   "/ensembles/consensuspresence/Bradypus variegatus_red.grd", sep="")
pres.raster <- raster(pres.file)
plot(pres.raster, breaks=c(-1, 0, 1), col=c("grey", "green"),
   main="after convex hull")
plot(mask.poly, add=T, border="blue")
par.old <- graphics::par(no.readonly=T)

# how distances were derived
# maximum distance between observations
ensemble.chull.buffer.distances(pres1, lonlat.dist=TRUE)
# the closest neighbour that is farthest away from each observation
# this is the distance calculated by MSDM_posteriori for buffer="species_specific"
ensemble.chull.buffer.distances(pres1, buffer.maxmins=TRUE, lonlat.dist=TRUE)

## End(Not run)
**Description**

The functions internally calls blockCV::spatialBlock and blockCV::envBlock. Syntax is very similar to that of BiodiversityR::ensemble.calibrate.weights.

**Usage**

```r
ensemble.spatialBlock(x = NULL, p = NULL, 
                      a = NULL, an = 1000, EPSG=NULL, 
                      excludep = FALSE, target.groups = FALSE, k = 4, 
                      factors = NULL, 
                      theRange = NULL, return.object = FALSE, ...)
```

```r
ensemble.envBlock(x = NULL, p = NULL, 
                   a = NULL, an = 1000, EPSG=NULL, 
                   excludep = FALSE, target.groups = FALSE, k = 4, 
                   factors = NULL, 
                   return.object = FALSE, ...)
```

**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, 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
- **EPSG**
  EPSG number ([https://spatialreference.org/](https://spatialreference.org/)) to be assigned internally to the coordinate reference system of the locations via `st_crs`. Although the function internally first assigns the coordinate reference from the RasterStack x via `crs`, this method fails in some situations as in the example shown below. In such cases, manually assigning the EPSG could resolve this problem.
- **excludep**
  parameter that indicates (if TRUE) that presence points will be excluded from the background points; see also `randomPoints`
- **target.groups**
  Parameter that indicates (if TRUE) that the provided background points (argument a) represent presence points from a target group sensu Phillips et al. 2009 (these are species that are all collected or observed using the same methods or equipment). Setting the parameter to TRUE results in selecting the centres of cells of the target groups as background points, while avoiding to select the same cells twice. Via argument excludep, it is possible to filter out cells with presence observations (argument p).
- **k**
  Integer value. The number of desired folds for cross-validation. The default is k = 4. The interpretation of the argument is exactly the same as in `ensemble.calibrate.models` and `kfold`.
- **factors**
  vector that indicates which variables are factors; see also `prepareData`
theRange Numeric value of the specified range by which blocks are created and training/testing data are separated. This distance should be in metres. See also spatialBlock.

return.object If TRUE, then also return ('block.object') the complete result of spatialBlock or envBlock. In addition (if TRUE), return the species data ('speciesData') that was created for blockCV. To visualize these results, see below or from foldExplorer.

... Other arguments to pass to spatialBlock or envBlock, such numLimit (The minimum number of points in each fold for training-presence, training-absence, testing-presence and testing-absence) and iteration (The number of attempts to create folds that fulfil the numLimit requirement).

Details

The functions internally call spatialBlock or envBlock.

The result of the function includes a list (k) with following elements. This list can be directly imported into ensemble.calibrate.weights, but only elements groupp and groupa will be used.

- p: Presence locations, created by ensemble.calibrate.models where points with missing data were excluded and possibly points were added for missing factor levels
- a: Background locations, created by ensemble.calibrate.models where points with missing data were excluded and possibly points were added for missing factor levels
- groupp: k-fold identities for the presence locations
- groupa: k-fold identities for the background locations

Optionally the function also returns elements block.object and speciesData. These can be used to visualize data with foldExplorer.

Value

The function returns a list with the following elements:

- k A list with data on folds that can be directly used by ensemble.calibrate.weights.
- block.object the results of spatialBlock or envBlock
- speciesData a SpatialPointsDataFrame with species data

Author(s)

Roeland Kindt (World Agroforestry Centre)

References

Examples

## Not run:

```r
library(blockCV)
library(sf)

# 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"))
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]

# choose background points
background <- randomPoints(predictors, n=1000, p=pres, excludep=T, extf=1.00)
background <- data.frame(background)
colnames(background)=c('lon', 'lat')

# spatial blocking with square blocks of 1000 km and minimum 20 points in each category
# fails if EPSG is not assigned
block.data <- ensemble.spatialBlock(x=predictors, p=pres, a=background,
                                     EPSG=NULL,
                                     showBlocks=F, theRange=1000000, k=4, numLimit=20, iteration=1000, return.object=T)
block.data <- ensemble.spatialBlock(x=predictors, p=pres, a=background,
                                     EPSG=4326,
                                     showBlocks=F, theRange=1000000, k=4, numLimit=20, iteration=1000, return.object=T)

# explore the results
foldExplorer(blocks=block.data$block.object, rasterLayer=predictors,
             speciesData=block.data$speciesData)

# apply in calibration of ensemble weights
# make sure that folds apply to subset of points
p.spatial <- block.data$k$p
a.spatial <- block.data$k$a
k.spatial <- block.data$k
ensemble.w1 <- ensemble.calibrate.weights(x=predictors,
                                          p=p.spatial, a=a.spatial, k=k.spatial,
                                          species.name="Bradypus",
                                          SINK=FALSE, PROBIT=TRUE,
                                          MAXENT=0, MAXNET=1, MAXLIKE=1, GBM=1, GBMSTEP=0, RF=0, CF=1,
                                          GLM=1, GLMSTEP=0, GAM=1, GAMSTEP=0, MGCV=0, MGCVFIX=0,
                                          return.object=T, showWeights=F)
```

ensemble.spatialThin

EARTH=0, RPART=0, NNET=1, FDA=0, SVM=0, SVME=0, GLMNET=0,
BIOLCI.0=1, BIOCLIN=1, DOMAIN=0, MAHAL=0, MAHAL01=0,
ENSEMBLE.tune=TRUE,
ENSEMBLE.best=0, ENSEMBLE.exponent=c(1, 2, 3),
ENSEMBLE.min=0.7,
Yweights="BIOMOD",
formulae.defaults=TRUE)

# confirm that correct folds were used
all.equal(ensemble.w1$groupp, block.data$k$groupp)
all.equal(ensemble.w1$groupa, block.data$k$groupa)

# environmental blocking with minimum 5 points in each category
block.data2 <- ensemble.envBlock(x=predictors, p=pres, a=background,
factors="biome",
k=4, numLimit=5, return.object=T)

# explore the results
foldExplorer(blocks=block.data2$block.object, rasterLayer=predictors,
speciesData=block.data2$speciesData)

# apply in calibration of ensemble weights
# make sure that folds apply to subset of points
p.env <- block.data2$k$p
a.env <- block.data2$k$a
k.env <- block.data2$k

ensemble.w2 <- ensemble.calibrate.weights(x=predictors,
  p=p.env, a=a.env, k=k.env,
species.name="Bradypus",
SINK=FALSE, PROBIT=TRUE,
MAXENT=0, MAXNET=1, MAXLIKE=1, GBM=1, GBMSTEP=0, RF=0, CF=1,
GLM=1, GLMSTEP=0, GAM=1, GAMSTEP=0, MGCV=0, MGCVFIX=0,
EARTH=0, RPART=0, NNET=1, FDA=0, SVM=0, SVME=0, GLMNET=0,
BIOLCI.0=1, BIOCLIN=1, DOMAIN=0, MAHAL=0, MAHAL01=0,
ENSEMBLE.tune=TRUE,
ENSEMBLE.best=0, ENSEMBLE.exponent=c(1, 2, 3),
ENSEMBLE.min=0.7,
factors="biome",
Yweights="BIOMOD",
formulae.defaults=TRUE)

# confirm that correct folds were used
all.equal(ensemble.w2$groupp, block.data2$k$groupp)
all.equal(ensemble.w2$groupa, block.data2$k$groupa)

## End(Not run)
**ensemble.spatialThin**  
Thinning of presence point coordinates in geographical or environmental space

**Description**

Function `ensemble.spatialThin` creates a randomly selected subset of point coordinates where the shortest distance (geodesic) is above a predefined minimum. The geodesic is calculated more accurately (via `distGeo`) than in the `spThin` or `red` packages.

**Usage**

```r
ensemble.spatialThin(x, thin.km = 0.1,  
runs = 100, silent = FALSE, verbose = FALSE,  
return.notRetained = FALSE)
```

```r
ensemble.spatialThin.quant(x, thin.km = 0.1,  
runs = 100, silent = FALSE, verbose = FALSE,  
LON.length = 21, LAT.length = 21)
```

```r
ensemble.environmentalThin(x, predictors.stack = NULL,  
extracted.data=NULL, thin.n = 50,  
runs = 100, pca.var = 0.95, silent = FALSE, verbose = FALSE,  
return.notRetained = FALSE)
```

```r
ensemble.environmentalThin.clara(x, predictors.stack = NULL, thin.n = 20,  
runs = 100, pca.var = 0.95, silent = FALSE, verbose = FALSE,  
clara.k = 100)
```

```r
ensemble.outlierThin(x, predictors.stack = NULL, k = 10,  
quant = 0.95, pca.var = 0.95,  
return.outliers = FALSE)
```

**Arguments**

- **x**
  Point locations provided in 2-column (lon, lat) format.
- **thin.km**
  Threshold for minimum distance (km) in final point location data set.
- **runs**
  Number of runs to maximize the retained number of point coordinates.
- **silent**
  Do not provide any details on the process.
- **verbose**
  Provide some details on each run.
- **return.notRetained**
  Return in an additional data set the point coordinates that were thinned out.
- **LON.length**
  Number of quantile limits to be calculated from longitudes; see also `quantile`.
- **LAT.length**
  Number of quantile limits to be calculated from latitudes; see also `quantile`.
- **predictors.stack**
  RasterStack object (stack) containing environmental layers that define the environmental space of point observations.
ensemble.spatialThin

extracted.data Data set with the environmental data at the point locations. If this data is provided, then this data will be used in the analysis and data will not be extracted from the predictors.stack.

thin.n Target number of environmentally thinned points.

pca.var Minimum number of axes based on the fraction of variance explained (default value of 0.95 indicates that at least 95 percent of variance will be explained on the selected number of axes). Axes and coordinates are obtained from Principal Components Analysis (scores).

clara.k The number of clusters in which the point coordinates will be divided by clara. Clustering is done in environmental space with point coordinates determined from Principal Components Analysis.

k The number of neighbours for the Local Outlier Factor analysis; see lof

quant The quantile probability above with local outlier factors are classified as outliers; see also quantile

return.outliers Return in an additional data set the point coordinates that were flagged as outliers.

Details

Locations with distances smaller than the threshold distance are randomly removed from the data set until no distance is smaller than the threshold. The function uses a similar algorithm as functions in the spThin or red packages, but the geodesic is more accurately calculated via distGeo.

With several runs (default of 100 as in the red package or some spThin examples), the (first) data set with the maximum number of records is retained.

Function ensemble.spatialThin.quant was designed to be used with large data sets where the size of the object with pairwise geographical distances could create memory problems. With this function, spatial thinning is only done within geographical areas defined by quantile limits of geographical coordinates.

Function ensemble.environmentalThin performs an analysis in environmental space similar to the analysis in geographical space by ensemble.spatialThin. However, the target number of retained point coordinates needs to be defined by the user. Coordinates are obtained in environmental space by a principal components analysis (function rda). Internally, first points are randomly selected from the pair with the smallest environmental distance until the selected target number of retained point coordinates is reached. From the retained point coordinates, the minimum environmental distance is determined. In a second step (more similar to spatial thinning), locations are randomly removed from all pairs that have a distance larger than the minimum distance calculated in step 1.

Function ensemble.environmentalThin.clara was designed to be used with large data sets where the size of the object with pairwise environmental distances could create memory problems. With this function, environmental thinning is done sequentially for each of the clusters defined by clara. Environmental space is obtained by by a principal components analysis (function rda). Environmental distances are calculated as the pairwise Euclidean distances between the point locations in the environmental space.

Function ensemble.outlierThin selects point coordinates that are less likely to be local outliers based on a Local Outlier Factor analysis (lof). Since LOF does not result in strict classification of outliers, a user-defined quantile probability is used to identify outliers.
Value

The function returns a spatially or environmentally thinned point location data set.

Author(s)

Roeland Kindt (World Agroforestry Centre)

References


See Also

ensemble.batch

Examples

```r
## Not run:
# get predictor variables, only needed for plotting
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]

# number of locations
nrow(pres)
par.old <- graphics::par(no.readonly=T)
par(mfrow=c(2,2))
pres.thin1 <- ensemble.spatialThin(pres, thin.km=100, runs=10, verbose=T)
plot(predictors[[1]], main="5 runs", ext=extent(SpatialPoints(pres.thin1)))
points(pres, pch=20, col="black")
points(pres.thin1, pch=20, col="red")
pres.thin2 <- ensemble.spatialThin(pres, thin.km=100, runs=10, verbose=T)
plot(predictors[[1]], main="5 runs (after fresh start)", ext=extent(SpatialPoints(pres.thin2)))
points(pres, pch=20, col="black")
points(pres.thin2, pch=20, col="red")
pres.thin3 <- ensemble.spatialThin(pres, thin.km=100, runs=100, verbose=T)
```

ensemble.spatialThin

```r
plot(predictors[[1]], main="100 runs", ext=extent(SpatialPoints(pres.thin3)))
points(pres, pch=20, col="black")
points(pres.thin3, pch=20, col="red")

pres.thin4 <- ensemble.spatialThin(pres, thin.km=100, runs=100, verbose=T)
plot(predictors[[1]], main="100 runs (after fresh start)", ext=extent(SpatialPoints(pres.thin4)))
points(pres, pch=20, col="black")
points(pres.thin4, pch=20, col="red")

graphics::par(par.old)

## thinning in environmental space

env.thin <- ensemble.environmentalThin(pres, predictors.stack=predictors, thin.n=60,
                                      return.notRetained=T)
pres.env1 <- env.thin$retained
pres.env2 <- env.thin$not.retained

# plot in geographical space
par.old <- graphics::par(no.readonly=T)
par(mfrow=c(1, 2))
plot(predictors[[1]], main="black = not retained", ext=extent(SpatialPoints(pres.thin3)))
points(pres.env2, pch=20, col="black")
points(pres.env1, pch=20, col="red")

# plot in environmental space
background.data <- data.frame(raster::extract(predictors, pres))
rdl.rda <- vegan::rda(X=background.data, scale=T)
# select number of axes
ax <- 2
while ( (sum(vegan::eigenvals(rda.rda.result)[c(1:ax)])/
         sum(vegan::eigenvals(rda.rda.result))) < 0.95 ) {ax <- ax+1}
rdl.scores <- data.frame(vegan::scores(rda.rda.result, display="sites", scaling=1, choices=c(1:ax)))
rownames(rdl.scores) <- rownames(pres)
points.in <- rdl.scores[which(rownames(rdl.scores) %in% rownames(pres.env1)), c(1:2)]
points.out <- rdl.scores[which(rownames(rdl.scores) %in% rownames(pres.env2)), c(1:2)]
plot(points.out, main="black = not retained", pch=20, col="black",
     xlim=range(rdl.scores[, 1]), ylim=range(rdl.scores[, 2]))
points(points.in, pch=20, col="red")

graphics::par(par.old)

## removing outliers
out.thin <- ensemble.outlierThin(pres, predictors.stack=predictors, k=10,
                                 return.outliers=T)
pres.out1 <- out.thin$inliers
pres.out2 <- out.thin$outliers

# plot in geographical space
par.old <- graphics::par(no.readonly=T)
par(mfrow=c(1, 2))
```

ensemble.zones

Mapping of environmental zones based on the Mahalanobis distance from centroids in environmental space.

Description

Function ensemble.zones maps the zone of each raster cell within a presence map based on the minimum Mahalanobis distance (via mahalanobis) to different centroids. Function ensemble.centroids defines centroids within a presence map based on Principal Components Analysis (via rda) and K-means clustering (via kmeans).

Usage

ensemble.zones(presence.raster = NULL, centroid.object = NULL, x = NULL, ext = NULL, RASTER.species.name = centroid.object$name, RASTER.stack.name = x@title, RASTER.format = "raster", RASTER.datatype = "INT1S", RASTER.NAflag = -127, KML.out = FALSE, KML.maxpixels = 100000, KML.blur = 10, CATCH.OFF = FALSE)

ensemble.centroids(presence.raster = NULL, x = NULL, categories.raster = NULL, an = 10000, ext = NULL, name = "Species001", pca.var = 0.95, centers = 0, use.silhouette = TRUE, plotit = FALSE, dev.new.width = 7, dev.new.height = 7)
Arguments

*presence.raster*  
RasterLayer object (*raster*) documenting presence (coded 1) of an organism

*centroid.object*  
Object listing values for centroids and covariance to be used with the *mahalanobis* distance (used internally by the prediction function called from *predict*).

*x*  
RasterStack object (*stack*) containing all environmental layers that correspond to explanatory variables

*ext*  
an Extent object to limit the predictions and selection of background points to a sub-region of *presence.raster* and *x*, typically provided as c(lonmin, lonmax, latmin, latmax). See also *randomPoints* and *extent*.

*RASTER.species.name*  
First part of the names of the raster file that will be generated, expected to identify the modelled species (or organism)

*RASTER.stack.name*  
Last part of the names of the raster file that will be generated, expected to identify the predictor stack used

*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 TRUE, then kml files will be saved in a subfolder 'kml/zones'.

*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*.

*CATCH.OFF*  
Disable calls to function *tryCatch*.

*categories.raster*  
RasterLayer object (*raster*) documenting predefined zones such as vegetation types. In case this object is provided, then centroids will be calculated for each zone.

*an*  
Number of presence points to be used for Principal Components Analysis (via *rda*); see also *prepareData* and *extract*

*name*  
Name for the centroid object, for example identifying the species and area for which centroids are calculated

*pca.var*  
Minimum number of axes based on the fraction of variance explained (default value of 0.95 indicates that at least 95 percent of variance will be explained on the selected number of axes). Axes and coordinates are obtained from Principal Components Analysis (*scores*).

*centers*  
Number of centers (clusters) to be used for K-means clustering (*kmeans*). In case a value smaller than 1 is provided, function *cascadeKM* is called to determine the optimal number of centers via the Calinski-Harabasz criterion.

*use.silhouette*  
If TRUE, then centroid values are only based on presence points that have silhouette values (*silhouette*) larger than 0.
plotit

If TRUE, then a plot is provided that shows the locations of centroids in geographical and environmental space. Plotting in geographical space is based on determination of the presence location (analogue) with smallest Mahalanobis distance to the centroid in environmental space.

dev.new.width

Width for new graphics device (dev.new). If < 0, then no new graphics device is opened.

dev.new.height

Height for new graphics device (dev.new). If < 0, then no new graphics device is opened.

Details

Function ensemble.zones maps the zone of each raster cell of a predefined presence map, whereby the zone is defined as the centroid with the smallest Mahalanobis distance. The function returns a RasterLayer object (raster) and possibly a KML layer.

Function ensemble.centroid provides the centroid locations in environmental space and a covariance matrix (cov) to be used with mahalanobis. Also provided is information on the analogue presence location that is closest to the centroid in environmental space.

Value

Function ensemble.centroid returns a list with following objects:

- centroids: Location of centroids in environmental space
- centroid.analogs: Location of best analogs to centroids in environmental space
- cov.mahal: Covariance matrix

Author(s)

Roeland Kindt (World Agroforestry Centre)

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)
predictors <- subset(predictors, subset=c("bio1", "bio5", "bio6", "bio7", "bio8",
  "bio12", "bio16", "bio17"))
predictors
predictors$title <- "base"

# choose background points
background <- randomPoints(predictors, n=1000, extf=1.00)

# predicted presence from GLM
ensemble.calibrate.step1 <- ensemble.calibrate.models(
  x=predictors, p=pres, a=background,
  species.name="Bradypus",
  MAXENT=0, MAXLIKE=0, GBM=0, GBMSTEP=0, RF=0, GLM=1, GLMSTEP=0,
  GAM=0, GAMSTEP=0, MGCV=0, MGCVFIX=0,
  EARTH=0, RPART=0, NNET=0, FDA=0, SVM=0, SVMFR=0, GLMNET=0,
  BIOCLIM.O=0, BIOCLIM=0, DOMAIN=0, MAHAL=0, MAHAL01=0,
  Yweights="BIOMOD",
  models.keep=TRUE)

ensemble.raster.results <- ensemble.raster(xn=predictors,
  models.list=ensemble.calibrate.step1$models,
  RASTER.species.name="Bradypus", RASTER.stack.name="base")

# get presence map as for example created with ensemble.raster in subfolder 'ensemble/presence'
# presence values are values equal to 1
presence.file <- paste("ensembles//presence//Bradypus_base.grd", sep="")
presence.raster <- raster(presence.file)

# let cascadeKM decide on the number of clusters
dev.new()
centroids <- ensemble.centroids(presence.raster=presence.raster,
  x=predictors, an=1000, plotit=T)
ensemble.zones(presence.raster=presence.raster, centroid.object=centroids,
  x=predictors, RASTER.species.name="Bradypus", KML.out=T)

dev.new()
zones.file <- paste("ensembles//zones//Bradypus_base.grd", sep="")
zones.raster <- raster(zones.file)
max.zones <- maxValue(zones.raster)
plot(zones.raster, breaks=c(0, c(1:max.zones)),
  col = grDevices::rainbow(n=max.zones), main="zones")
ensemble.zones(presence.raster=presence.raster, centroid.object=centroids,
  x=predictors, RASTER.species.name="Bradypus", KML.out=T)

# manually choose 6 zones
dev.new()
centroids6 <- ensemble.centroids(presence.raster=presence.raster,
  x=predictors, an=1000, plotit=T, centers=6)
ensemble.zones(presence.raster=presence.raster, centroid.object=centroids6,
  x=predictors, RASTER.species.name="Bradypus6", KML.out=T)

dev.new()
zones.file <- paste("ensembles//zones//Bradypus6_base.grd", sep="")
zones.raster <- raster(zones.file)
max.zones <- maxValue(zones.raster)
plot(zones.raster, breaks=c(0, c(1:max.zones)),
  col = grDevices::rainbow(n=max.zones), main="six zones")
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

```r
evaluation.strip.data(xn = NULL, ext = NULL, models.list = NULL, input.weights = models.list$output.weights, steps=200, CATCH.OFF = FALSE)
```

```r
evaluation.strip.plot(data, TrainData=NULL, variable.focal = NULL, model.focal = NULL, ylim=c(0, 1.25), dev.new.width = 7, dev.new.height = 7, ...
```

Arguments

- **xn**: RasterStack object (stack) containing all layers that correspond to explanatory variables of an ensemble calibrated earlier with `ensemble.calibrate.models`. 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.calibrate.weights` can be used.
- **steps**: number of steps within the range of a continuous explanatory variable.
- **CATCH.OFF**: Disable calls to function `tryCatch`.
- **data**: data set with ranges of environmental variables and fitted suitability models, typically returned by `evaluation.strip.data`.
- **TrainData**: Data set representing the calibration data set. If provided, then a boxplot will be added for presence locations via `boxplot`.
variable.focal  focal explanatory variable for plots with evaluation strips
model.focal    focal model for plots with evaluation strips
ylim          range of Y-axis
dev.new.width Width for new graphics device (dev.new). If < 0, then no new graphics device
               is opened.
dev.new.height Heigth for new graphics device (dev.new). If < 0, then no new graphics device
               is opened.
...            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 en-
vironmental variables encountered in the RasterStack object (x) and the suitability modelling ap-
proaches 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 num-
ber 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

Kindt R. 2018. Ensemble species distribution modelling with transformed suitability values. Envi-

method for plotting predicted responses from species distribution models. Ecological Modelling
186: 280-289

See Also

ensemble.calibrate.models and 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"))
predictors <- stack(predictors)
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
background <- randomPoints(predictors, n=1000, 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
# MAXLIKE not included as does not allow predictions for data.frames
# ENSEMBLE.min and ENSEMBLE.weight.min set very low to explore all
# algorithms.
# If focus is on actual ensemble, then set ENSEMBLE.min and
# ENSEMBLE.weight.min to more usual values
ensemble.calibrate <- ensemble.calibrate.models(x=predictors,
        p=pres_train, a=backg_train,
        pt=pres_test, at=backg_test,
        ENSEMBLE.min=0.5, ENSEMBLE.weight.min = 0.001,
        MAXENT=0, MAXNET=1, MAXLIKE=1, GBM=1, GBMSTEP=0, RF=1, CF=1,
        GLM=1, GLMSTEP=1, GAM=1, GAMSTEP=1, MGCV=1, MGCVFIX=1,
        EARTH=1, RPART=1, NN=1, FDA=1, SVM=1, SVM=1,
        BIOCLIM.O=1, BIOCLIM=1, DOMAIN=1, MAHAL=0, MAHAL01=1,
        Yweights="BIOMOD",
        PLOTS=FALSE, models.keep=TRUE)

# obtain data for plotting the evaluation strip
strip.data <- evaluation.strip.data(xn=predictors, steps=500,
        models.list=ensemble.calibrate$models)
# in case predictions for DOMAIN failed
# however, ENSEMBLE should also be recalculated
DOMAIN.model <- ensemble.calibrate$models$DOMAIN
strip.data$plot.data[, "DOMAIN"] <- dismo::predict(object=DOMAIN.model, x=strip.data$plot.data)

# in case predictions for MAHAL01 failed
predict.MAHAL01 <- function(model, newdata, MAHAL.shape) {
  p <- dismo::predict(object=model, x=newdata)
  p <- p - 1 - MAHAL.shape
  p <- abs(p)
  p <- MAHAL.shape / p
  return(as.numeric(p))
}

MAHAL01.model <- ensemble.calibrate$models$MAHAL01
MAHAL.shape1 <- ensemble.calibrate$models$formulae$MAHAL.shape
strip.data$plot.data[, "MAHAL01"] <- predict.MAHAL01(model=MAHAL01.model, newdata=strip.data$plot.data, MAHAL.shape=MAHAL.shape1)

# create graphs
evaluation.strip.plot(data=strip.data$plot.data, variable.focal="bio6", TrainData=strip.data$TrainData, type="o", col="red")
evaluation.strip.plot(data=strip.data$plot.data, model.focal="ENSEMBLE", TrainData=strip.data$TrainData, type="o", col="red")

## End(Not run)

faramea

Faramea 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 Faramea 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(faramea)

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 Tcm Tgo Tl
Farameaoccidentalis 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).

References


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://ifri.forgov.org/

Examples

data(ifri)

<table>
<thead>
<tr>
<th>importancevalue</th>
<th>Importance Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Importance Value</td>
<td></td>
</tr>
</tbody>
</table>

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)
importancevalue

References


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')

# Calculate diversity profiles from importance values
imp <- importancevalue.comp(ifri, site='plotID', species='species',
   count='count', basal='basal', factor='forest')
vals <- imp[["values"]]
for (i in 1:length(vals)) {
   imp.i <- data.frame(imp[[vals[i]]])
   name.i <- paste(vals[i], ".Renyi", sep="")
   imp[[name.i]] <- renyi(imp.i$importance.value)
}

# LOT more diverse
imp$LOT.Renyi - imp$MCF.Renyi
imp$LOT.Renyi - imp$YSF.Renyi

# YSF and MCF different richness and evenness
imp$YSF.Renyi - imp$MCF.Renyi
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)
stackcommunitydataset(comm, remove.zeroes=FALSE, order.sites=FALSE, order.species=FALSE)

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)

comm Community data set

remove.zeroes Should rows with zero abundance be removed?

order.sites Should sites be ordered alphabetically?

order.species Should species be ordered alphabetically?

Details

makecommunitydataset 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.

stackcommunitydataset reverses the actions of makecommunitydataset and recreates the data in stacked format.

Value

The function provides a community dataset from another data frame.

Author(s)

Roeland Kindt (World Agroforestry Centre)

References


https://www.worldagroforestry.org/output/tree-diversity-analysis

Examples

```r
## Not run:
dune.file <- normalizePath(paste(system.file(package="BiodiversityR"),
   '/etc/dunestacked.csv', sep=''))
dune.stacked <- read.csv(dune.file)

# dune.stacked has different variables for sites, species and abundance
head(dune.stacked)
dune.comm2 <- makecommunitydataset(dune.stacked, row='sites', column='species',
   value='abundance')

# recreate the original stack
dune.stacked2 <- stackcommunitydataset(dune.comm2, remove.zeroes=T)
```
multiconstrained  

### End(Not run)

## multiconstrained  

Pairwise Comparisons for All Levels of a Categorical Variable by RDA, CCA or Capscale

### Description

This function implements pairwise comparisons for categorical variable through `capscale`, `cca`, `dbrda` 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

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

### Arguments

- `method`  
  Method for constrained ordination analysis; one of "rda", "cca", "dbrda" or "capscale".

- `formula`  
  Model formula as in `capscale`, `cca` or `rda`. The LHS can be a community data matrix or a distance matrix for `capscale`.

- `data`  
  Data frame containing the variables on the right hand side of the model formula as in `capscale`, `cca` or `rda`.

- `distance`  
  Dissimilarity (or distance) index in `vegdist` used if the LHS of the formula is a data frame instead of dissimilarity matrix; used only with function `vegdist` and partial match to "manhattan", "euclidean", "canberra", "bray", "kuleczynski", "jaccard", "gower", "morisita", "horn" or "mountford". This argument is only used for `capscale` 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 `capscale`. 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 `capscale`.

- `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 `anova.cca` results for all contrasts is provided.

- `...`  
  Other parameters passed to `anova.cca`.
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 categoricl 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


Examples

```r
## 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, data=dune.env,
    distance="bray", add=TRUE, contrast=3)
## End(Not run)
```
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", "horns" 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


Examples

```r
## 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**

```r
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


Examples

```r
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

Calculate the NNET Result with the Smallest Value from Various Random Starts

Description

This function provides the best solution from various calls to the \texttt{nnet} feed-forward artificial neural networks function (\texttt{nnet}).

Usage

\texttt{nnetrandom(formula, data, tries=10, leave.one.out=F,...)}

Arguments

\begin{itemize}
  \item \texttt{formula} \hspace{1cm} Formula as passed to \texttt{nnet}.
  \item \texttt{data} \hspace{1cm} Data as passed to \texttt{nnet}.
  \item \texttt{tries} \hspace{1cm} Number of calls to \texttt{nnet} to obtain the best solution.
  \item \texttt{leave.one.out} \hspace{1cm} Calculate leave-one-out predictions.
  \item \texttt{...} \hspace{1cm} Other arguments passed to \texttt{nnet}.
\end{itemize}

Details

This function makes various calls to \texttt{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 \texttt{nnet}, but adds the following components:

\begin{itemize}
  \item \texttt{range} \hspace{1cm} Summary of the observed "values".
  \item \texttt{tries} \hspace{1cm} Number of different attempts to iterate an ANN.
  \item \texttt{CV} \hspace{1cm} Predicted class when not using the respective sample unit for iterating ANN.
  \item \texttt{successful} \hspace{1cm} Test whether leave-one-out statistics provided the same class as the original class.
\end{itemize}

Author(s)

Roeland Kindt (World Agroforestry Centre)
Examples

```r
## 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 function investigates the relationship between the species vectors and the position of sites on an ordination axis. A GAM (\texttt{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


https://www.worldagroforestry.org/output/tree-diversity-analysis

Examples

```r
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)
```

```r
ordisymbol
Add Other Graphical Items to Ordination Diagrams
```

Description

Functions to add some other graphical items to ordination diagrams than provided within \texttt{vegan} by \texttt{ordihull}, \texttt{ordispider}, \texttt{ordiarrows}, \texttt{ordisegments}, \texttt{ordigrid}, \texttt{ordiellipse}, \texttt{ordicluster} and \texttt{lines.spantree}.

Usage

```r
ordisymbol(ordiplot, y, factor, col = 1, colors = TRUE, pchs = TRUE,
    rainbow_hcl = TRUE, rainbow_hcl.c = 90, rainbow_hcl.l = 50,
    rainbow = TRUE, heat.colors = FALSE, terrain.colors = FALSE,
    topo.colors = FALSE, cm.colors = FALSE,
    legend = TRUE, legend.x = "topleft", legend.ncol = 1, ...)
ordibubble(ordiplot, var,...)
```
ordisymbol

ordicluster2(ordiplot, cluster, mingroups = 1, maxgroups = nrow(ordiplot$sites), ...)  
ordinearest(ordiplot, dist,...)  
ordivector(ordiplot, spec, lty=2,...)

Arguments

ordiplot            An ordination graph created by ordiplot (vegan).
y                  Environmental data frame.
factor              Variable of the environmental data frame that defines subsets to be given different symbols.
var                  Continous variable of the environmental dataset or species from the community dataset.
col                  Colour (as points).
colors               Apply different colours to different factor levels
pchs                Apply different symbols (plotting characters) to different factor levels (as in points)
rainbow_hcl         Use rainbow_hcl colours (rainbow_hcl)
rainbow_hcl.c       Set the chroma value
rainbow_hcl.l       Set the luminance value
rainbow            Use rainbow colours
heat.colors         Use heat colours
terrain.colors      Use terrain colours
topo.colors         Use topo colours
cm.colors           Use cm colours
legend              Add the legend.
legend.x             Location of the legend; see also legend.
legend.ncol       the number of columns in which to set the legend items; see also legend
cluster             Cluster object.
mingroups           Minimum of clusters to be plotted.
maxgroups           Maximum of clusters to be plotted.
dist                Distance matrix.
spec                Species name from the community dataset.
lty               Line type as specified for par.
...                 Other arguments passed to functions points, symbols, ordihull or arrows.

Details

Function ordisymbol plots different levels of the specified variable in different symbols and different colours. In case more than one colour palettes are selected, the last palette selected will be used.

Function ordibubble draws bubble diagrams indicating the value of the specified continuous variable. Circles indicate positive values, squares indicate negative values.
Function `ordicluster2` provides an alternative method of overlaying information from hierarchical clustering on an ordination diagram than provided by function `ordicluster`. 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 diagramme 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**


[https://www.worldagroforestry.org/output/tree-diversity-analysis](https://www.worldagroforestry.org/output/tree-diversity-analysis)

**Examples**

```r
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)
```
**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**

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

**Arguments**

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

**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


https://www.worldagroforestry.org/output/tree-diversity-analysis

Examples

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

```r
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 `plot.radfit`.

**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:

radfit results of `radfit`.
fisherfit results of `fisherfit`.
prestonfit results of `prestonfit`.

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

Author(s)

Roeland Kindt (World Agroforestry Centre)

References


https://www.worldagroforestry.org/output/tree-diversity-analysis

Examples

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

---

### rankabundance

#### Rank Abundance Curves

**Description**

Provides methods of calculating rank-abundance curves.

**Usage**

```r
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), srt=0, ...)

rankabuncomp(x, y="", factor, return.data=T, specnames=c(1:3), 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.
srt  The string rotation in degrees of the species names (as in par).
return.data  Return the data used for plotting.
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 plot or points.

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 (plower, pupper) and the proportion of species ranks (in percentage).

Author(s)

Roeland Kindt (World Agroforestry Centre)

References


https://www.worldagroforestry.org/output/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(1:3))
rankabunplot(RankAbun.1, scale='logabun', addit=FALSE, specnames=c(1:30),
            srt=45, ylim=c(1,100))
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.
## Not run:
# ggplot2 plotting method
# Only label the two most abundant species
RA.data <- rankabuncomp(dune, y=dune.env, factor='Management',
                        return.data=TRUE, specnames=c(1:2), legend=FALSE)
library(ggplot2)
library(ggrepel)

# possibly need for extrafont::loadfonts(device="win") to have Arial
# as alternative, use library(ggThemeAssist)
BioR.theme <- theme(
  panel.background = element_blank(),
  panel.border = element_blank(),
  panel.grid = element_blank(),
  axis.line = element_line("gray25"),
  panel.text = element_text(face="bold"),
  panel.grid.major = element_line(color="gray20", linetype="solid"),
  panel.grid.minor = element_line(color="gray20", linetype="solid"),
  axis.title = element_text(family="Arial", size=12, color="black"),
  axis.text = element_text(family="Arial", size=12, color="black"),
  axis.ticks = element_blank(),
  legend.text = element_text(family="Arial", size=12, color="black"),
  legend.title = element_text(family="Arial", size=12, color="black"),
  legend.position = "bottom",
  legend.key = element_blank(),
  legend.title.position = "top")

## END NOT RUN
text = element_text(size = 12, family="Arial"),
axis.text = element_text(size = 10, colour = "gray25"),
axis.title = element_text(size = 14, colour = "gray25"),
legend.title = element_text(size = 14),
legend.text = element_text(size = 14),
legend.key = element_blank())

plotgg1 <- ggplot(data=RA.data, aes(x = rank, y = abundance)) +
  scale_x_continuous(expand=c(0, 1), sec.axis = dup_axis(labels=NULL, name=NULL)) +
  scale_y_continuous(expand=c(0, 1), sec.axis = dup_axis(labels=NULL, name=NULL)) +
  geom_line(aes(colour=Grouping), size=1) +
  geom_point(aes(colour=Grouping, shape=Grouping), size=5, alpha=0.7) +
  geom_text_repel(data=subset(RA.data, labelit == TRUE),
                  aes(colour=Grouping, label=species),
                  angle=45, nudge_x=1, nudge_y=1, show.legend=FALSE) +
  BioR.theme +
  scale_color_brewer(palette = "Set1") +
  labs(x = "rank", y = "abundance", colour = "Management", shape = "Management")

plotgg1

# use different facets
# now label first 10 species
RA.data <- rankabuncomp(dune, y=dune.env, factor='Management',
                        return.data=TRUE, specnames=c(1:10), legend=FALSE)

plotgg2 <- ggplot(data=RA.data, aes(x = rank, y = abundance)) +
  scale_x_continuous(expand=c(0, 1), sec.axis = dup_axis(labels=NULL, name=NULL)) +
  scale_y_continuous(expand=c(0, 1), sec.axis = dup_axis(labels=NULL, name=NULL)) +
  geom_line(aes(colour=Grouping), size=1) +
  geom_point(aes(colour=Grouping), size=5, alpha=0.7) +
  geom_text_repel(data=subset(RA.data, labelit == TRUE),
                  aes(label=species),
                  angle=45, nudge_x=1, nudge_y=1, show.legend=FALSE) +
  BioR.theme +
  scale_color_brewer(palette = "Set1") +
  facet_wrap(~ Grouping) +
  labs(x = "rank", y = "abundance", colour = "Management")

plotgg2

## End(Not run) # dontrun

---

**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

```r
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)
```

import.with.readxl(file = file.choose(), data.type = "community", sheet = NULL,
sitenames = "sites", column = "species", value = "abundance",
factor = "", level = ",", cepnames = FALSE,
write.csv = FALSE, csv.file = paste(data.type, ".csv", sep=""))
```

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.
- `file` Location of the Excel (or Access) file.
- `data.type` Type of the data set to be imported: one of "community", "environmental" or "stacked".
- `sheet` Name of the sheet of the Excel file to import from (if missing, then `data.type` is used)
- `sitenames` Name of categorical variable that provides the names for the sites.
- `column` Name of the categorical variable for the columns of the crosstabulation (typically indicating species); passed to `makecommunitydataset`. 

removeNAcomm

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; passed to `makecommunitydataset`.

cepnames Should the names of columns be abbreviated via `make.cepnames` (TRUE) or not (FALSE).

write.csv Create a comma-delimited text file in the working directory (if TRUE).

csv.file Name of the comma-delimited text file to be created.

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 `replaceNAcomm` substitutes cells containing NA with 0 in the community data set.

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.

Function `import.with.readxl` provides methods of importing community or environmental datasets through `read_excel`.

For stacked datasets, a community data set is created with function `makecommunitydataset`. For community data with more species than the limited number of columns in Excel, this may be the only option of importing a community dataset.
An additional advantage of the function is that the community and environmental data can be stored in the same file.

You may want to check compatibility of the community and environmental datasets with functions `check.datasets` and modify the community dataset through `same.sites`.

**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**


[https://www.worldagroforestry.org/output/tree-diversity-analysis](https://www.worldagroforestry.org/output/tree-diversity-analysis)

**Examples**

```r
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)
```

### renyiresult

**Alternative Renyi Diversity Results**

**Description**

Provides some alternative methods of obtaining results on Renyi diversity profile values than provided by `renyi` (vegan).
Usage

renyiresult(x, y=NULL, factor, level, method = "all",
scales = c(0, 0.25, 0.5, 1, 2, 4, 8, Inf), evenness = FALSE,...)

renyiplot(xr, addit=FALSE, pch = 1,
xlab = "alpha", ylab = "H-alpha", ylim = NULL,
labelit = TRUE, legend = TRUE, legend.x="topleft", legend.ncol = 8,
col = 1, cex = 1, rainbow = TRUE, evenness = FALSE, ...)

renyiaccumresult(x, y=NULL, 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 = FALSE,...)

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 renyiresult.
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).
legend.x Location of the legend; see also legend.
legend.ncol number of columns for the legend (as in function legend).
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.
**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 `renyicomp` 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 `renyicomp` calculates the diversity profile for all levels of a selected environmental variable separately.

Functions `renyicomp` 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**


https://www.worldagroforestry.org/output/tree-diversity-analysis
https://rpubs.com/Roeland-KINDT

See Also

renyi.long, renyicomp.long

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

## Not run:
# ggplot2 plotting method
Renyi.2 <- renyicomp(dune, y=dune.env, factor='Management',
   scales=c(0, 0.25, 0.5, 1, 2, 4, 8, Inf), permutations=100, plotit=F)
Renyi.2
library(ggplot2)

# change the theme
# possibly need for extrafont::loadfonts(device="win") to have Arial
# as alternative, use library(ggThemeAssist)
BioR.theme <- theme(
   panel.background = element_blank(),
   panel.border = element_blank(),
   panel.grid = element_blank(),
   axis.line = element_line("gray25"),
   text = element_text(size = 12, family="Arial"),
   axis.text = element_text(size = 10, colour = "gray25"),
   axis.title = element_text(size = 14, colour = "gray25"),
   legend.title = element_text(size = 14),
   legend.text = element_text(size = 14),
   legend.key = element_blank())

renyi.long2 <- renyicomp.long(Renyi.2, label.freq=1)

plotgg1 <- ggplot(data=renyi.long2, aes(x=Scales, y=Diversity, ymax=UPR, ymin=LWR)) +
   scale_x_discrete() +
   scale_y_continuous(sec.axis = dup_axis(labels=NULL, name=NULL)) +
   geom_line(data=renyi.long2, aes(x=Obs, colour=Grouping), size=2) +
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 `glm` or `glm.nb`.
- **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 `gam` (if "TRUE") or by `surf.ls` (if "FALSE").
- **npol**: Degree of polynomial surface as passed to `surf.ls`.
- **plotit**: Plot the interpolated surface (through `interp` and the residuals).
- **filled**: Fill the contours by `filled.contour`.
- **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


[https://www.worldagroforestry.org/output/tree-diversity-analysis](https://www.worldagroforestry.org/output/tree-diversity-analysis)

Examples

```r
library(vegan)
library(mgcv)
library(akima)
data(faramea)
Count.model1 <- lm(Faramea.occidentalis ~ Precipitation,
                  data=faramea, na.action=na.exclude)
```
sites.long <- residualssurface(Count.model1, na.omit(faramea),
  'UTM.EW', 'UTM.NS', gam=TRUE, plotit=TRUE, bubble=TRUE)

sites.long

Helper Functions to Prepare Plotting of Accumulation, Diversity Profile and Ordiplot Results via ggplot2

Description

These functions organize outputs from `ordiplot`, `accumcomp` and `renyicomp` so these can be plotted with `ggplot`.

Usage

sites.long(x, env.data = NULL)

species.long(x, spec.data = NULL)

centroids.long(y, grouping, FUN = mean, centroids.only = FALSE)

vectorfit.long(z)

ordisurfgrid.long(z)

ordiellipse.long(z, grouping.name = "Grouping")

pvclust.long(cl, cl.data)

axis.long(w, choices = c(1, 2), cmdscale.model=FALSE, CAPdiscrim.model=FALSE)

accumcomp.long(x, ci = 2, label.freq = 1)

renyicomp.long(x, label.freq = 1)

renyi.long(x, env.data=NULL, label.freq = 1)

Arguments

x Result of `ordiplot`, `accumcomp` or `renyicomp`

env.data Environmental descriptors for each site.

spec.data Descriptors for each species.

y Result of function `sites.long`.

grouping Variable defining the centroids

FUN A function to compute the summary statistics which can be applied to all data subsets, as in `aggregate`
centroids.only  Return the coordinates for the centroids
z  Result of `envfit`, `ordisurf` or `ordiellipse`
grouping.name  Name for the categorical variable, expected as the factor used in the earlier ordiellipse call.
cl  Result of `pvclust`
cl.data  Result of `ordicluster`
w  Ordination object from which the `ordiplot` was obtained, expected to be fitted in vegan.
choices  Ordination axes selected, as in `ordiplot`
cmdscale.model  Use TRUE is the model was fitted via `cmdscale`
CAPdiscrim.model  Use TRUE is the model was fitted via `CAPdiscrim`
ci  Multiplier for confidence intervals as in `specaccum`. In case 'NA' is provided, then the multiplier is calculated via `qt`.
label.freq  Frequency of labels along the x-axis (count between labels).

Details

Examples for ordiplot results are shown below.

Function `pvclust.long` combines data from `pvclust` with coordinates of nodes and branches from `ordicluster`. The variable of prune allows to remove higher levels of nodes and branches in the clustering hierarchy in a similar way as argument prune for `ordicluster` - see examples.

See also section: see examples for species accumulation curves and Renyi diversity profiles

Value

These functions produce data.frames that can subsequentially be plotted via `ggplot` methods.

Author(s)

Roeland Kindt

References


https://www.worldagroforestry.org/output/tree-diversity-analysis
https://rpubs.com/Roeland-KINDT

See Also

`accumcomp`, `renyicomp`
Examples

## Not run:

```r
# ggplot2 plotting method
library(ggplot2)
library(ggforce)
library(concaveman)
library(ggrepel)
library(ggsci)
library(dplyr)
library(vegan)
data(dune)
data(dune.env)
attach(dune)
attach(dune.env)

Ordination.model1 <- capscale(dune ~ Management, data=dune.env,
   distance='kulczynski', sqrt.dist=F, add='cailliez')

plot1 <- ordiplot(Ordination.model1, choices=c(1,2), scaling='species')

# obtain 'long' data from the ordiplot object
sites1 <- sites.long(plot1, env.data=dune.env)
species1 <- species.long(sites1)
centroids1 <- centroids.long(sites1, Management, FUN=median)

centroids2 <- centroids.long(sites1, Management, FUN=median, centroids.only=TRUE)

# information on percentage variation from the fitted ordination
axislabs <- axis.long(Ordination.model1, choices=c(1, 2))

# change the theme
# possibly need for extrafont::loadfonts(device="win") to have Arial
# as alternative, use library(ggThemeAssist)
BioR.theme <- theme(
   panel.background = element_blank(),
   panel.border = element_blank(),
   panel.grid = element_blank(),
   axis.line = element_line("gray25"),
   text = element_text(size = 12, family="Arial"),
   axis.text = element_text(size = 10, colour = "gray25"),
   axis.title = element_text(size = 14, colour = "gray25"),
   legend.title = element_text(size = 14),
   legend.text = element_text(size = 14),
   legend.key = element_blank())

# no species scores
# centroids calculated directly via the centroids.long function
plotgg1 <- ggplot() +
   geom_vline(xintercept = c(0), color = "grey70", linetype = 2) +
   geom_hline(yintercept = c(0), color = "grey70", linetype = 2) +
```

```r
```
xlab(axislabs[1, "label"]]) +
ylab(axislabs[2, "label"]]) +
scale_x_continuous(sec.axis = dup_axis(labels=NULL, name=NULL)) +
scale_y_continuous(sec.axis = dup_axis(labels=NULL, name=NULL)) +
geom_mark_hull(data=sites1, aes(x=axis1, y=axis2, colour = Management),
concavity = 0.1, alpha=0.8, size=0.2, show.legend=FALSE) +
geom_point(data=sites1, aes(x=axis1, y=axis2, colour=Management, shape=Management),
size=5) +
# geom_segment(data=species1, aes(x=0, y=0, xend=axis1*2, yend=axis2*2),
# size=1.2, arrow=arrow()) +
# geom_label_repel(data=species1, aes(x=axis1*2, y=axis2*2, label=labels)) +
geom_point(data=centroids.long(sites1, grouping=Management, centroids.only=TRUE),
aes(x=axis1c, y=axis2c, colour=Centroid, shape=Centroid), size=10, show.legend=FALSE) +
geom_segment(data=centroids.long(sites1, grouping=Management),
aes(x=axis1c, y=axis2c, xend=axis1, yend=axis2, colour=Management),
size=1, show.legend=FALSE) +
BioR.theme +
ggsci::scale_colour_npg() +
coord_fixed(ratio=1)

plotgg1

# select species to plot based on goodness of fit
spec.envfit <- envfit(plot1, env=dune)
spec.data1 <- data.frame(r=spec.envfit$vectors$r, p=spec.envfit$vectors$pvals)
species2 <- species.long(plot1, spec.data=spec.data1)
species2 <- species2[species2$r > 0.6, ]

# after_scale introduced in ggplot2 3.3.0
plotgg2 <- ggplot() +
  geom_vline(xintercept = c(0), color = "grey70", linetype = 2) +
  geom_hline(yintercept = c(0), color = "grey70", linetype = 2) +
  xlab(axislabs[1, "label"]]) +
  ylab(axislabs[2, "label"]]) +
scale_x_continuous(sec.axis = dup_axis(labels=NULL, name=NULL)) +
scale_y_continuous(sec.axis = dup_axis(labels=NULL, name=NULL)) +
geom_mark_ellipse(data=sites1, aes(x=axis1, y=axis2,
  colour=Management, fill=after_scale(alpha(colour, 0.2))),
  expand=0, size=0.2, show.legend=TRUE) +
geom_point(data=sites1, aes(x=axis1, y=axis2, colour=Management, shape=Management),
  size=5) +
geom_segment(data=centroids.long(sites1, grouping=Management),
aes(x=axis1c, y=axis2c, xend=axis1, yend=axis2, colour=Management),
  size=1, show.legend=FALSE) +
geom_segment(data=species2, aes(x=0, y=0, xend=axis1*2, yend=axis2*2),
  size=1.2, arrow=arrow()) +
geom_label_repel(data=species2, aes(x=axis1*2, y=axis2*2, label=labels)) +
BioR.theme +
ggsci::scale_colour_npg() +
coord_fixed(ratio=1)

plotgg2
# Add contour and vector for a continuous explanatory variable
Ordination.model2 <- capscale(dune ~ Management, data=dune.env,
  distance='kulczynski', sqrt.dist=F, add='cailliez')

plot2 <- ordiplot(Ordination.model2, choices=c(1,2), scaling='species')
sites2 <- sites.long(plot2, env.data=dune.env)
axislabs <- axis.long(Ordination.model2, choices=c(1, 2))
dune.envfit <- envfit(plot2, dune.env)
vectors2 <- vectorfit.long(dune.envfit)
A1.surface <- ordisurf(plot2, y=A1)
A1.surface
A1.grid <- ordisurfgrid.long(A1.surface)

plotgg3 <- ggplot() +
  geom_contour_filled(data=A1.grid, aes(x=x, y=y, z=z)) +
  geom_vline(xintercept = c(0), color = "grey70", linetype = 2) +
  geom_hline(yintercept = c(0), color = "grey70", linetype = 2) +
  xlab(axislabs[1, "label"])
  ylab(axislabs[2, "label"])
  scale_x_continuous(sec.axis = dup_axis(labels=NULL, name=NULL)) +
  scale_y_continuous(sec.axis = dup_axis(labels=NULL, name=NULL)) +
  geom_point(data=sites2, aes(x=axis1, y=axis2, size=A1), shape=21, colour="black", fill="red") +
  geom_segment(data=subset(vectors2, vector=A1), aes(x=0, y=0, xend=axis1*2, yend=axis2*2),
    size=1.2, arrow=arrow()) +
  geom_label_repel(data=subset(vectors2, vector=A1), aes(x=axis1*2, y=axis2*2,
    label=vector), size=5) +
  BioR.theme +
  scale_fill_viridis_d() +
  scale_size(range=c(1, 20)) +
  labs(fill="A1") +
  coord_fixed(ratio=1)

plotgg3

# after_stat introduced in ggplot2 3.3.0
plotgg4 <- ggplot() +
  geom_contour(data=A1.grid, aes(x=x, y=y, z=z, colour=factor(after_stat(level))), size=2) +
  geom_vline(xintercept = c(0), color = "grey70", linetype = 2) +
  geom_hline(yintercept = c(0), color = "grey70", linetype = 2) +
  xlab(axislabs[1, "label"])
  ylab(axislabs[2, "label"])
  scale_x_continuous(sec.axis = dup_axis(labels=NULL, name=NULL)) +
  scale_y_continuous(sec.axis = dup_axis(labels=NULL, name=NULL)) +
  geom_point(data=sites2, aes(x=axis1, y=axis2, size=A1), shape=21, colour="black", fill="red") +
  geom_label_repel(data=sites2, aes(x=axis1, y=axis2, label=labels),
    colour='black', size=4) +
  BioR.theme +
  scale_colour_viridis_d() +
  scale_size(range=c(1, 20)) +
  labs(colour="A1") +

plotgg4
coord_fixed(ratio=1)

plotgg4

# example of Voronoi segmentation
plotgg5 <- ggplot(data=sites2, aes(x=axis1, y=axis2)) +
  geom_voronoi_tile(aes(fill = Management, group=-1L), max.radius=0.2) +
  geom_voronoi_segment(colour="grey50") +
  geom_vline(xintercept = c(0), color = "grey70", linetype = 2) +
  geom_hline(yintercept = c(0), color = "grey70", linetype = 2) +
  xlab(axislabs[1, "label"])
  ylab(axislabs[2, "label"])
  scale_x_continuous(sec.axis = dup_axis(labels=NULL, name=NULL)) +
  scale_y_continuous(sec.axis = dup_axis(labels=NULL, name=NULL)) +
  geom_point() +
  BioR.theme +
  ggsci::scale_colour_npg() +
  coord_fixed(ratio=1)

plotgg5

# adding ellipse via ordiellipse
plot3 <- ordiplot(Ordination.model1, choices=c(1,2), scaling='species')
axislabs <- axis.long(Ordination.model1, choices=c(1, 2))
Management.ellipses <- ordiellipse(plot3, groups=Management, display="sites", kind="sd")
Management.ellipses.long2 <- ordiellipse.long(Management.ellipses, grouping.name="Management")

plotgg6 <- ggplot() +
  geom_vline(xintercept = c(0), color = "grey70", linetype = 2) +
  geom_hline(yintercept = c(0), color = "grey70", linetype = 2) +
  xlab(axislabs[1, "label"])
  ylab(axislabs[2, "label"])
  scale_x_continuous(sec.axis = dup_axis(labels=NULL, name=NULL)) +
  scale_y_continuous(sec.axis = dup_axis(labels=NULL, name=NULL)) +
  geom_polygon(data=Management.ellipses.long2,
                aes(x=axis1, y=axis2, colour=Management,
                    fill=after_scale(alpha(colour, 0.2))),
                size=0.2, show.legend=FALSE) +
  geom_point(data=sites1, aes(x=axis1, y=axis2, colour=Management, shape=Management),
             size=5) +
  geom_segment(data=centroids.long(sites1, grouping=Management),
               aes(x=axis1c, y=axis2c, xend=axis1, yend=axis2, colour=Management),
               size=1, show.legend=FALSE) +
  BioR.theme +
  ggsci::scale_colour_npg() +
  coord_fixed(ratio=1)

plotgg6

# adding cluster results via pvclust.long
library(pvclust)
# transformation as pvclust works with Euclidean distance
dune.Hellinger <- disttransform(dune, method = "hellinger")
dune.pv <- pvclust(t(dune.Hellinger),
                     method.hclust = "mcquitty",
                     method.dist = "euclidean",
                     nboot = 1000)

plot(dune.pv)
pvrect(dune.pv, alpha = 0.89, pv = "au")

# Model fitted earlier
plot1 <- ordiplot(Ordination.model1, choices = c(1, 2), scaling = "species")
cl.data1 <- ordicluster(plot1, cluster = as.hclust(dune.pv$hclust))
sites1 <- sites.long(plot1, env.data = dune.env)
axislabs <- axis.long(Ordination.model1, choices = c(1, 2))
cl.data1 <- ordicluster(plot2, cluster = as.hclust(dune.pv$hclust))
pvlong <- pvclust.long(dune.pv, cl.data1)

# as in example for ordicluster, prune higher level hierarchies
plotgg7 <- ggplot() +
  geom_vline(xintercept = c(0), color = "grey70", linetype = 2) +
  geom_hline(yintercept = c(0), color = "grey70", linetype = 2) +
  xlab(axislabs[1, "label"])
  ylab(axislabs[2, "label"])
  scale_x_continuous(sec.axis = dup_axis(labels = NULL, name = NULL)) +
  scale_y_continuous(sec.axis = dup_axis(labels = NULL, name = NULL)) +
  geom_segment(data = subset(pvlong$segments, pvlong$segments$prune > 3),
               aes(x = x1, y = y1, xend = x2, yend = y2, colour = au >= 0.89,
                    size = au),
               show.legend = TRUE) +
  geom_point(data = subset(pvlong$nodes, pvlong$nodes$prune > 3),
             aes(x = x, y = y, fill = au >= 0.89),
             shape = 21, size = 2, colour = "black") +
  geom_point(data = sites1,
             aes(x = axis1, y = axis2, shape = Management),
             colour = "darkerolivegreen4", alpha = 0.9, size = 5) +
  geom_text(data = sites1,
             aes(x = axis1, y = axis2, label = labels)) +
  BioR.theme +
  ggsci::scale_colour_npg() +
  scale_size(range = c(0.3, 2)) +
  scale_shape_manual(values = c(15, 16, 17, 18)) +
  guides(shape = guide_legend(override.aes = list(linetype = 0))) +
  coord_fixed(ratio = 1)

plotgg7

## End(Not run) # dontrun
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

```r
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 vertix 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


https://www.worldagroforestry.org/output/tree-diversity-analysis

Examples

```r
library(splancs)
area <- array(c(10,10,15,35,40,35,35,10), dim=c(6,2))
landuse1 <- array(c(10,10,15,15,35,30,35,30,35,30,35,10), dim=c(4,2))
landuse2 <- array(c(10,10,15,15,35,30,35,30,35,30,35,15), 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

```r
data(transfgradient)
```
transfspecies

Format
A data frame with 19 observations on the following variable.

gradient  a numeric vector

Source

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)}
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

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)}

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


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


Examples

data(warenv)
Index

* datasets
  BCI.env, 10
  faramea, 116
  ifri, 117
  transfgradient, 155
  transfspecies, 156
  warcom, 157
  warenv, 163

* multivariate
  accumresult, 4
  add.spec.scores, 7
  balanced.specaccum, 9
  BiodiversityRGUI, 12
  CAPdiscrim, 13
  caprescale, 16
  crosstabanalysis, 17
  deviancepercentage, 18
  dist.eval, 20
  dist.zeros, 21
  distdisplayed, 22
  disttransform, 24
  diversityresult, 25
  importancevalue, 118
  loaded.citations, 121
  makecommunitydataset, 121
  multiconstrained, 123
  nested.anova.dbдра, 125
  NMSrandom, 126
  nnetrandom, 128
  ordicoeno, 129
  ordisymbol, 130
  PCAsignificance, 133
  radfitresult, 134
  rankabundance, 135
  removeNAcom, 138
  renyiresult, 141
  residualssurface, 145
  sites.long, 147
  spatialsample, 154

* package
  BiodiversityR-package, 3
  .packages, 121
  accumcomp, 147, 148
  accumcomp(accumresult), 4
  accumcomp.long, 6
  accumcomp.long(sites.long), 147
  accumplot(accumresult), 4
  accumresult, 4, 9, 10, 140
  add.spec.scores, 7, 15
  addLayer, 38, 60
  adonis, 125
  aggregate, 147
 anova.cca, 123, 124
 anova.glм, 19
 anova.negbin, 19
  aoo, 96
  areaPolygon, 90, 91
  arrows, 131, 133
  as.dist, 13
  axis.long(sites.long), 147
  balanced.specaccum, 9
  BCI, 10, 117
  BCI.env, 10, 117
  bioclim, 38, 45, 47, 59
  BiodiversityR (BiodiversityR-package), 3
  BiodiversityR-package, 3
  BiodiversityR.changeLog, 11
  BiodiversityRGUI, 3, 12
  bioenv, 20
  biovars, 74, 75
  boxplot, 113
  CAPdiscrim, 13, 148
  caprescale, 16
  capscale, 13, 16, 123–125
  cascadeKM, 110
  cca, 123, 124
INDEX

centroids.long (sites.long), 147
cforest, 37, 39, 59, 61
cforest_control, 39, 61
check.datasets, 141
check.datasets (removeNAscomm), 138
check.ordiscores (removeNAscomm), 138
chisq.test, 17, 18
circles, 35, 57
citation, 121
clara, 106
cmdscale, 8, 14, 16, 148
cor, 8, 23, 57
cov, 30, 111
coverscale, 24
crosstabanalysis, 17
crs, 101
dataType, 29, 36, 73, 80, 89, 96, 110
dbrda, 123
decostand, 24
dev.new, 41, 111, 114
deviancepercentage, 18
dispweight, 24
dist.eval, 20
dist.zeroes, 21
distconnected, 20
distdisplayed, 22
distGeo, 105, 106
distrtransform, 24
diversity, 25, 26
diversitycomp (diversityresult), 25
diversityresult, 25, 140
diversityvariables (diversityresult), 25
domain, 38, 59
earth, 37, 40, 59, 62
eccrop, 74
ensemble.accepted.categories
  (ensemble.dummy.variables), 70
ensemble.analogue, 28
ensemble.analogue.object, 29
ensemble.area (ensemble.raster), 88
ensemble.batch, 32, 66, 77, 84, 92, 97, 107
ensemble.bioclim, 38, 40, 45, 50, 59, 62, 81
ensemble.bioclim.graph, 47, 49, 50, 81
ensemble.bioclim.object, 45, 49
ensemble.calibrate.models, 33, 42, 43, 52, 71, 89, 91, 92, 101, 102, 113, 114
ensemble.calibrate.weights, 33, 35, 42, 43, 92, 102
ensemble.centroids (ensemble.models), 52
ensemble.chull.buffer.distances
  (ensemble.red), 95
ensemble.chull.create (ensemble.red), 95
ensemble.chull.MSDM (ensemble.red), 95
ensemble.drop1
  (ensemble.calibrate.models), 52
ensemble.dummy.variables, 70
ensemble.ecocrop, 73
ensemble.ecocrop.object, 73
ensemble.envBlock
  (ensemble.spatialBlock), 100
ensemble.environmentalThin
  (ensemble.spatialThin), 105
ensemble.evaluate, 76
ensemble.formulae, 38, 60
ensemble.formulae
  (ensemble.calibrate.models), 52
ensemble.habitat.change
  (ensemble.raster), 88
ensemble.mean (ensemble.batch), 32
ensemble.novel, 31, 46, 47, 50, 79
ensemble.novel.object, 80
ensemble.outlierThin
  (ensemble.spatialThin), 105
ensemble.pairs
  (ensemble.calibrate.models), 52
ensemble.PET.seasons, 83, 87
ensemble.PET.seasons, 85
ensemble.plot (ensemble.batch), 32
ensemble.prec.season
  (ensemble.PET.seasons), 85
ensemble.raster, 33, 42, 43, 57, 65, 66, 71, 81, 88, 111, 114
ensemble.red, 95
ensemble.season.suitability
  (ensemble.PET.seasons), 85
ensemble.SEDI (ensemble.evaluate), 76
ensemble.simplified.categories
  (ensemble.dummy.variables), 70
ensemble.spatialBlock, 100
ensemble.spatialThin, 33, 42, 104
ensemble.strategy
(ensemble.calibrate.models), 52
ensemble.threshold, 76
ensemble.threshold
(ensemble.calibrate.models), 52
ensemble.tmean.season
(ensemble.PET.seasons), 85
ensemble.VIF, 36, 42
ensemble.VIF
(ensemble.calibrate.models), 52
ensemble.weights
(ensemble.calibrate.models), 52
ensemble.zones, 109
envBlock, 102
evf
estimator, 26
evaluate, 57, 58, 63, 65, 76, 77, 90
evaluation.strip.data, 113
evaluation.strip.plot, 91, 92
evaluation.strip.plot
(evaluation.strip.data), 113
extent, 110, 113
extract, 29, 38, 41, 46, 56, 57, 60, 71, 90, 95, 101, 110
faramea, 116
fda, 38, 40, 59, 62
filled.contour, 146
fisher.alpha, 25, 26
fisherfit, 134, 135
foldExplorer, 102
freq, 70, 71
gam, 23, 37, 40, 59, 61, 62, 129, 130, 145, 146
gam.control, 38, 60
gbm, 37, 39, 59, 60
gbm.step, 37, 39, 59, 61, 62, 64
gBuffer, 96
get.block, 35, 62
getCrop, 74, 86
ggplot, 147
glm, 19, 37–40, 59–62, 146
glm.control, 38, 60
glm.nb, 19, 146
glmnet, 38, 40, 59, 62
ifri, 117, 120
import.with.readxl(removeNAcomm), 138
importancevalue, 118
initMDS, 127
interp, 146
isoMDS, 8, 127
kfold, 35, 36, 57, 62, 64, 101
kmeans, 109, 110
KML, 29, 36, 46, 73, 74, 80, 90, 110
ksvm, 38, 40, 59, 62
lda, 14
legend, 129, 131, 142
levels, 114
lines.spantree, 130, 132
loaded.citations, 121
lof, 106
mahal, 38, 40, 60, 62, 64, 91
mahalanobis, 30, 109–111
make.cepnames, 140
makecommunitydataset, 121, 139, 140
mantel, 23
map.sdm, 95
mask, 80
maxent, 37–39, 56–58, 60
maxlike, 37, 39, 59, 60
maxnet, 37, 39, 58, 60
mean, 114
mess, 81
metaMDS, 8, 126, 127
ModelEvaluation, 91
multiconstrained, 123
na.omit, 19
nested.anova.dbrda, 125
nested.npmanova(nested.anova.dbrda), 125
nicheOverlap, 42
NMSrandom, 8, 126
nnet, 38, 40, 59, 60, 62–64, 128
nnetrandom, 128
optim, 39, 60
optimal.thresholds, 36, 58, 65
ordiarrows, 130
ordiburble (ordisymbol), 130
ordiclient, 130, 132, 148
ordiclient2 (ordisymbol), 130
ordicoeno, 129
ordiellipse, 130, 148
ordiellipse.long (sites.long), 147
ordiequilibriumcircle
    (PCAsignificance), 133
ordigrid, 130
ordihull, 130, 131
ordinearest (ordisymbol), 130
ordiplot, 15, 16, 22, 23, 129, 131, 133, 147, 148
ordisegments, 130
ordispider, 130
ordisurf, 148
ordisurfgrid.long (sites.long), 147
ordisymbol, 130
ordivector (ordisymbol), 130
pairs, 65
par, 5, 129, 131, 136
PCAsignificance, 133
plot, 5, 42, 114, 129, 136, 143
plot.radfit, 134, 135
plot.specaccum, 4–6, 9
pointDistance, 96
points, 5, 129, 131, 136, 142
postMDS, 8
predict, 28, 29, 34, 45, 73, 79, 80, 89, 110, 113
predict.glmnet, 40, 62, 64
predict.maxnet, 39, 60
prepare.bioenv (dist.eval), 20
prepareData, 38, 41, 46, 56, 57, 60, 62, 71, 90, 95, 101, 110
prestonfit, 134, 135
pvclust, 148
pvclust.long (sites.long), 147
qnorm, 46
qt, 148
quantile, 30, 46, 80, 105, 106
radfit, 134, 135
radfitresult, 134
rainbow_hcl, 131
randomForest, 37, 39, 59, 61
randomPoints, 30, 35, 39, 56, 57, 60, 101, 110, 113
rankabuncomp (rankabundance), 135
rankabundance, 135
rankabunplot (rankabundance), 135
raster, 70, 71, 80, 90, 95, 96, 110, 111
rda, 7, 8, 106, 109, 110, 123, 124, 133
read_excel, 140
removeNACcomm, 138
removeNAenv (removeNACcomm), 138
removezerospecies (removeNACcomm), 138
renyi, 141–143
renyi.long, 144
renyi.long (sites.long), 147
renyiaccum, 143
renyiaccumresult (renyiresult), 141
renyicomponent, 147, 148
renyicomponent (renyiresult), 141
renyicomponent.long, 144
renyicomponent.long (sites.long), 147
renyiplot (renyiresult), 141
renyiresult, 140, 141
replaceNACcomm (removeNACcomm), 138
residuals, 146
residualssurface, 145
round, 63
rpart, 37, 40, 59, 62
rpart.control, 40, 62
s, 40, 62
same.sites, 141
same.sites (removeNACcomm), 138
sammon, 8, 127
scores, 106, 110, 140
silhouette, 110
sink, 36, 57, 90
sites.long, 147, 147
spatialBlock, 102
SpatialPointsDataFrame, 102
spatialsample, 154
specaccum, 4–6, 9, 10, 143, 148
species.long (sites.long), 147
specnumber, 25, 26
specpool, 25, 26
ssb, 35, 57
st_crs, 101
stack, 29, 34, 45, 50, 56, 73, 80, 84, 86, 89, 101, 105, 110, 113
stackcommunitydataset (makecommunitydataset), 121
step.gam, 37, 40, 59, 61
stepAIC, 37, 40, 59, 61
subsetcomm (removeNACcomm), 138
surf.is, 145, 146
svm, 38, 40, 59, 62
symbols, 131
threshold, 36, 57, 58, 76
title, 5
transfgradient, 155, 156
transfspecies, 155, 156
trunc, 91
tryCatch, 29, 36, 46, 57, 74, 80, 84, 86, 90, 110, 113

vectorfit.long (sites.long), 147
vegdist, 14, 20–23, 123, 125, 127
vif, 57, 63

warcom, 157
warenv, 163
wascores, 8
writeFormats, 29, 36, 46, 73, 80, 89, 96, 110
writeRaster, 29, 36, 46, 71, 73, 80, 84, 86, 89, 96, 110
wrld_simpl, 42