Package 'modEvA'

September 25, 2026

Type Package

Title Model Evaluation and Analysis

Version 3.47

Date 2026-09-25

Maintainer A. Marcia Barbosa <ana.marcia.barbosa@gmail.com>

Imports effectsize, graphics, grDevices, stats, terra (> 1.5-50)

Description Analyses species distribution models and evaluates their performance. It includes functions for variation partitioning, extracting variable importance, computing several metrics of model discrimination and calibration performance, optimizing prediction thresholds based on a number of criteria, performing multivariate environmental similarity surface (MESS) analysis, and displaying various analytical plots. Initially described in Barbosa et al. (2013) <doi:10.1111/ddi.12100>.

LazyLoad yes

LazyData yes

License GPL-3

URL http://modeva.r-forge.r-project.org/

BugReports https://github.com/AMBarbosa/modEvA/issues

NeedsCompilation no

Author A. Marcia Barbosa [aut, cre],
Jennifer A. Brown [aut],
Alberto Jimenez-Valverde [aut],
Raimundo Real [aut],
Oswald van Ginkel [ctb],
Jurica Levatic [ctb],
Victoria Formoso-Freire [ctb],
Andres Baselga [ctb],
Carola Gomez-Rodriguez [ctb],
Carlos C. Ramos [ctb],
Jose Carlos Guerrero [fnd]

Depends R (>= 2.10)

Repository CRAN

Date/Publication 2026-09-25 11:20:02 UTC

Contents

modEvA-package . . . 3
applyThreshold . . . 4
arrangePlots . . . 7
AUC . . . 8
Boyce . . . 13
confusionLabel . . . 16
confusionMatrix . . . 18
Dsquared . . . 21
errorMeasures . . . 23
evaluate . . . 25
evenness . . . 27
getBins . . . 28
getModEqn . . . 31
getThreshold . . . 32
HLfit . . . 35
inputMunch . . . 39
logLike . . . 40
lollipop . . . 42
MESS . . . 44
Miller . . . 47
MillerCalib . . . 50
mod2obspred . . . 52
modEvAmethods . . . 53
multModEv . . . 54
OA . . . 56
optiPair . . . 57
optiThresh . . . 60
plotCoeffs . . . 63
plotGLM . . . 65
poMeasures . . . 67
predDensity . . . 70
predPlot . . . 72
prevalence . . . 75
pseudoRsq . . . 76
ptsrast2obspred . . . 79
quantReclass . . . 80
range01 . . . 82
RMSE . . . 83
rotif.mods . . . 84
RsqGLM . . . 85
similarity . . . 86
standard01 . . . 89
threshMeasures . . . 90
varImp . . . 96
varPart . . . 100

Index **105**

modEvA-package *Model Evaluation and Analysis*

Description

The modEvA package can analyse species distribution models and evaluate their performance. It includes functions for performing variation partitioning; calculating several measures of model discrimination, classification, explanatory power, and calibration; optimizing prediction thresholds based on a number of criteria; performing multivariate environmental similarity surface (MESS) analysis; and displaying various analytical plots.

Details

Package:	modEvA
Type:	Package
Version:	3.47
Date:	2026-09-25
License:	GPL-3

Author(s)

Barbosa A.M., Brown J.A., Jimenez-Valverde A., Real R.

A. Marcia Barbosa <ana.marcia.barbosa@gmail.com>

References

Barbosa A.M., Real R., Munoz A.R. & Brown J.A. (2013) New measures for assessing model equilibrium and prediction mismatch in species distribution models. Diversity and Distributions 19: 1333-1338 (DOI: 10.1111/ddi.12100)

See Also

PresenceAbsence, **ROCR**, **verification**, **Metrics**

Examples

```
# load sample models:
data(rotif.mods)

# choose a particular model to play with:
mod <- rotif.mods$models[[1]]

# plot this model:
plotGLM(model = mod)
```

```
# compute the Root Mean Square Error and other error measures:
errorMeasures(model = mod)

# extract variable importance from the model:
varImp(model = mod)
varImp(model = mod, imp.type = "perm")

# compute the area under the ROC curve for the model:
AUC(model = mod)
AUC(model = mod, curve = "PR")

# compute some threshold-based measures for this model:
threshMeasures(model = mod, thresh = 0.5)
threshMeasures(model = mod, thresh = "preval")
threshMeasures(model = mod, thresh = "maxTSS")

# compute similarity (also threshold-based) measures:
similarity(model = mod, thresh = "maxTSS")

# compute optimal thresholds based on several criteria:
optiThresh(model = mod, measures = c("CCR", "Sensitivity", "kappa", "TSS"),
ylim = c(0, 1), pch = 20, cex = 0.5)

# compute the optimal threshold balancing two evaluation measures:
optiPair(model = mod, measures = c("Sensitivity", "Specificity"))

# compute model calibration (reliability) metrics:
HLfit(model = mod, bin.method = "n.bins")
Miller(model = mod)
Boyce(model = mod)
Dsquared(model = mod)
pseudoRsq(model = mod)

# compute a bunch of evaluation measures for a set of models:
multModEv(models = rotif.mods$models[1:4], thresh = "preval",
bin.method = "quantiles")
```

`applyThreshold` *Apply threshold(s) to model predictions*

Description

This function applies a threshold value to the continuous predictions of a model, converting them to binary predictions: 1 for values above the threshold, and 0 for values below it. If two thresholds are provided (e.g. to separate high, low and intermediate predictions), the result is 0 below the lowest threshold, 1 above the highest threshold, and 0.5 between them.

Usage

```
applyThreshold(model = NULL, obs = NULL, pred = NULL, thresh, pbg = FALSE,
right = FALSE, interval = 0.01, quant = 0, na.rm = TRUE, verbosity = 2)
```

Arguments

model	optional binary-response model object of class "glm", "gam", "gbm", "randomForest" or "bart". If this argument is provided, 'obs' and 'pred' will be extracted with mod2obspred. Alternatively, you can input the 'obs' and 'pred' arguments instead of 'model'.
obs	alternatively to 'model' and together with 'pred', a numeric vector of observed presences (1) and absences (0) of a binary response variable. Alternatively (and if 'pred' is a 'SpatRaster'), a two-column matrix or data frame containing, respectively, the x (longitude) and y (latitude) coordinates of the presence points, in which case the 'obs' vector will be extracted with ptsrast2obspred. This argument is ignored if 'model' is provided.
pred	alternatively to 'model' and together with 'obs', a vector with the corresponding predicted values of presence probability, habitat suitability, environmental favourability or alike. Must be of the same length and in the same order as 'obs'. Alternatively (and if 'obs' is a set of point coordinates), a 'SpatRaster' map of the predicted values for the entire evaluation region, in which case the 'pred' vector will be extracted with ptsrast2obspred. This argument is ignored if 'model' is provided.
thresh	numeric vector of length 1 or 2, containing the threshold value(s) with which to reclassify 'pred', or the criteria under which to compute these thresholds – run modEvAmethods("getThreshold") for available options, and see Details in getThreshold for their description.
pbg	logical value to pass to inputMunch indicating whether to use presence/background (rather than presence/absence) data. Default FALSE.
right	logical value indicating if the interval should be closed on the right (and open on the left) or vice versa, i.e., if predictions equalling the threshold value(s) should be classified as lower rather than higher. The default is FALSE.
interval	Argument to pass to optiThresh indicating the interval between the thresholds to test, if 'thresh' implies optimizing a threshold-based measure. The default is 0.01. Smaller values may provide more precise results but take longer to compute.
quant	Numeric value indicating the proportion of presences to discard if any of 'thresh' is "MTP" (minimum training presence). With the default value 0, MTP will be the threshold at which all observed presences are classified as such; with e.g. quant=0.05, MTP will be the threshold at which 5% presences will be classified as absences.
na.rm	Logical value indicating whether NA values should be ignored. Defaults to TRUE.
verbosity	integer value indicating the amount of messages to print. Defaults to 2, for the maximum amount of messages.

Details

Several criteria have been proposed for selecting thresholds with which to convert continuous model predictions (of presence probability, habitat suitability or alike) into binary predictions of presence or absence. A threshold is required for computing threshold-based model evaluation metrics, such as those in threshMeasures. This function reclassifies the predictions of a model given one or two numeric thresholds, or one or two threshold selection criteria implemented in getThreshold.

Value

This function returns an object of the same class as 'pred' with the reclassified values after application of the threshold.

Author(s)

A. Marcia Barbosa

See Also

getThreshold, threshMeasures

Examples

```
# load sample models:
data(rotif.mods)

# choose a particular model to play with:
mod <- rotif.mods$models[[1]]

applyThreshold(model = mod, thresh = "maxTSS")


# you can also use applyThreshold with vectors of observed and predicted values:

presabs <- mod$y
prediction <- mod$fitted.values

applyThreshold(pred = prediction, thresh = 0.5)

applyThreshold(pred = prediction, thresh = c(0.2, 0.8))

applyThreshold(pred = prediction, thresh = "meanPred")

applyThreshold(obs = presabs, pred = prediction, thresh = "preval")

applyThreshold(obs = presabs, pred = prediction, thresh = "MTP")

applyThreshold(obs = presabs, pred = prediction, thresh = "MTP",
quant = 0.05)


# 'obs' can also be a table of presence point coordinates
```

```
# and 'pred' a SpatRaster of predicted values
```

`arrangePlots` *Arrange plots*

Description

Get an appropriate row/column combination (for `par(mfrow)`) for arranging a given number of plots within a plotting window.

Usage

```
arrangePlots(n.plots, landscape = FALSE)
```

Arguments

`n.plots`	number of plots to be placed in the graphics device.
`landscape`	logical, whether the plotting window should be landscape/horizontal (number of columns larger than the number of rows) or not. The value does not make a difference if the number of plots makes for a square plotting window.

Details

This function is used internally by `optiThresh`, but can also be useful outside it.

Value

An integer vector of the form c(nr, nc) indicating, respectively, the number of rows and of columns of plots to set in the graphics device.

Author(s)

A. Marcia Barbosa

See Also

`plot`, `layout`

Examples

```
arrangePlots(10)

arrangePlots(10, landscape = TRUE)


# a more practical example:

data(iris)
```

```
names(iris)

# say you want to plot all columns in a nicely arranged plotting window:

par(mfrow = arrangePlots(ncol(iris)))

for (i in 1:ncol(iris)) {
  plot(1:nrow(iris), iris[, i])
}
```

AUC *Area Under the Curve*

Description

This function calculates the Area Under the Curve of the receiver operating characteristic (ROC) plot, or alternatively the precision-recall (PR) plot, for either a model object or two matching vectors of observed binary (1 for occurrence vs. 0 for non-occurrence) and predicted continuous (e.g. occurrence probability) values, respectively.

Usage

```
AUC(model = NULL, obs = NULL, pred = NULL, simplif = FALSE,
interval = "auto", FPR.limits = c(0, 1), curve = "ROC", partial = FALSE,
max.omiss = 0.05, pbg = FALSE, method = NULL, plot = TRUE, diag = TRUE,
diag.col = "lightblue3", diag.lty = 2, curve.col = "steelblue4", curve.lty = 1,
curve.lwd = 2, plot.values = TRUE, plot.digits = 3, plot.preds = FALSE,
grid = FALSE, grid.lty = 1, xlab = "auto", ylab = "auto", cex.lab = 0.9,
ticks = FALSE, na.rm = TRUE, rm.dup = FALSE, verbosity = 2, ...)
```

Arguments

model	optional binary-response model object of class "glm", "gam", "gbm", "randomForest" or "bart". If this argument is provided, 'obs' and 'pred' will be extracted with mod2obspred. Alternatively, you can input the 'obs' and 'pred' arguments instead of 'model'.
obs	alternatively to 'model' and together with 'pred', a numeric vector of observed presences (1) and absences (0) of a binary response variable. Alternatively (and if 'pred' is a 'SpatRaster'), a two-column matrix or data frame containing, respectively, the x (longitude) and y (latitude) coordinates of the presence points, in which case the 'obs' vector will be extracted with ptsrast2obspred. This argument is ignored if 'model' is provided.
pred	alternatively to 'model' and together with 'obs', a vector with the corresponding predicted values of presence probability, habitat suitability, environmental favourability or alike. Must be of the same length and in the same order as 'obs'. Alternatively (and if 'obs' is a set of point coordinates), a 'SpatRaster' map of the predicted values for the entire evaluation region, in which case the 'pred' vector will be extracted with ptsrast2obspred. This argument is ignored if 'model' is provided.

simplif	logical value (default FALSE) indicating whether to use a faster version that returns only the AUC value (and the plot if 'plot = TRUE'). Used internally by other functions in the package.
FPR.limits	(NOT YET IMPLEMENTED) numerical vector of length 2 indicating the limits of false positive rate between which to calculate a partial AUC. The default is c(0, 1), for considering the whole AUC. Pending implementation. Meanwhile, you can try e.g. the roc function in the **pROC** package.
curve	character indicating whether to compute the "ROC" (receiver operating characteristic) or the "PR" (precision-recall) curve.
partial	[EXPERIMENTAL, UNDER CONSTRUCTION!] (if curve="ROC") logical value (default FALSE) indicating whether to compute a (standardized) partial AUC under a given omission threshold (Peterson et al. 2008).
max.omiss	(if partial=TRUE and curve="ROC") numeric value between 0 and 1 (default 0.05) defining the threshold omission rate for the partial AUC.
pbg	logical value to pass to inputMunch indicating whether to use presence/background (rather than presence/absence) data. Default FALSE.
interval	interval of threshold values at which to compute and plot the true- and false-positive rates. Can be a (preferably small) numeric value between 0 and 1; or the new default "auto", which will be either 0.01 (old default) or 0.001, depending on whether 'pred' does or doesn't have values in all thirds of the [0, 1] range. Larger intervals provide faster computation, but may generate innacurate curves especially if 'pred' has too few values towards one of the extremes (e.g. towards 1). Note that, if method = "rank" (the default if curve = "ROC"), 'interval' does not affect the obtained AUC value (although it can visually affect the size of the plotted curve, especially when there are very few 'pred' values towards 0 or towards 1), as the AUC is calculated with the Mann-Whitney-Wilcoxon statistic and is therefore threshold-independent. If method != "rank" (or, by extension, if curve = "PR" – see 'method' argument), smaller 'interval' values will provide more accurate AUC values. Smaller 'interval' values also improve the output 'meanPrecision', as this is averaged across all threshold values.
method	character indicating with which method to calculate the AUC value. Available options are "rank" (the default and most accurate, but implemented only if curve = "ROC") and "trapezoid" (the default if curve = "PR"). The latter may be computed more accurately if 'interval' is smaller (see 'interval' argument).
plot	logical, whether or not to plot the curve. Defaults to TRUE.
diag	logical, whether or not to add the reference diagonal (if plot = TRUE). Defaults to TRUE.
diag.col	line colour for the reference diagonal (if diag = TRUE).
diag.lty	line type for the reference diagonal (if diag = TRUE).
curve.col	line colour for the curve.
curve.lty	line type for the curve.
curve.lwd	line width for the curve.
plot.values	logical, whether or not to show in the plot the values associated to the curve (e.g., the AUC). Defaults to TRUE.

plot.digits	integer number indicating the number of digits to which the values in the plot should be rounded. Defaults to 3. This argument is ignored if 'plot' or 'plot.values' are set to FALSE.
plot.preds	logical value indicating whether the proportions of 'pred' values for each threshold should be plotted as proportionally sized blue circles. Can also be provided as a character vector specifying if the circles should be plotted on the "curve" (the default) and/or at the "bottom" of the plot. The default is FALSE for no circles, but it may be interesting to try it, especially if your curve has long straight lines or does not cover the full length of the plot.
grid	logical, whether or not to add a grid to the plot, marking the analysed thresholds. Defaults to FALSE.
grid.lty	line type for the grid (if grid = TRUE).
xlab	label for the x axis. By default, a label is automatically generated according to the specified 'curve'.
ylab	label for the y axis. By default, a label is automatically generated according to the specified 'curve'.
cex.lab	'cex' (amount by which the text should be magnified) for the axis labels. Default 0.9.
ticks	logical, whether or not to add blue tick marks at the bottom of the plot to mark the thresholds at which there were values from which to to draw the curve. Defaults to FALSE.
na.rm	Logical value indicating if missing values should be ignored in computations. The default is TRUE.
rm.dup	If TRUE and if 'pred' is a SpatRaster and if there are repeated points within the same pixel, a maximum of one point per pixel is used to compute the presences. See examples in ptsrast2obspred. The default is FALSE.
verbosity	integer specifying the amount of messages to display. Defaults to the maximum implemented; lower numbers (down to 0) decrease the number of messages.
...	further arguments to be passed to the plot function.

Details

In the case of the "ROC" curve (the default), the AUC is a measure of the overall discrimination power of the predictions, or the probability that an occurrence site has a higher predicted value than a non-occurrence site. It can thus be calculated with the Wilcoxon rank sum statistic, as is done with the default method="rank". There's also an option to compute, instead of the ROC curve, the precision-recall ("PR") curve, which is more robust to imbalanced data, e.g. species rarity (Sofaer et al. 2019), as it doesn't value true negatives. Mind that, while the baseline for a random AUC-ROC is 0.5, for AUC-PR there is no fixed reference; see Precision-Recall-Gain curves (Flach & Kull 2015) for an alternative, implemented e.g. in the 'prg' package (https://github.com/meeliskull/prg).

If 'curve' is set to "PR", or if 'method' is manually set to "trapezoid", the AUC value will be more accurate if 'interval' is decreased (see 'method' and 'interval' arguments above). The plotted curve will also be more accurate with smaller 'interval' values, especially for imbalanced datasets (which can cause an apparent disagreement between the look of the curve and the actual value of the AUC).

Swets (1988) proposed the following thresholds for interpreting AUC-ROC scores:

- below 0.6 = fail
- 0.6 to 0.7 = poor
- 0.7 to 0.8 = fair
- 0.8 to 0.9 = good
- 0.9 or higher = excellent

Mind that the ROC AUC has been widely criticized (e.g. Lobo et al. 2008, Jimenez-Valverde et al. 2013), though is still a widely used metric in model evaluation. It is highly correlated with species prevalence (as are the generality of discrimination and classification metrics, including TSS), so prevalence is also output by the AUC function (if `simplif = FALSE`, the default) for reference.

Although there are functions to compute the AUC in other R packages (e.g. **ROCR**, **PresenceAbsence**, **verification**, **Epi**, **PRROC**, **PerfMeas**, **precrec**), the `AUC` function is more compatible (in terms of arguments and outputs) with the remaining functions in **modEvA**; and it can be applied not only to a set of observed vs. predicted values, but also directly to a model object of class "glm", "gam", "gbm", "randomForest" or "bart", or to a set of presence point coordinates and a raster map of the predictions in the model evaluation region.

Value

If `simplif = TRUE`, the function returns only the AUC value (a numeric value between 0 and 1). Otherwise (the default), it returns a `list` with the following components:

`thresholds`	a data frame of the true and false positives, the sensitivity, specificity and recall of the predictions, and the number of predicted values at each analysed threshold.
`N`	the total number of obervations.
`prevalence`	the proportion of presences (i.e., ones) in the data (which correlates with the AUC of the "ROC" plot).
`AUC`	the value of the AUC).
`AUCratio`	the ratio of the obtained AUC value to the null expectation (0.5).
`meanPrecision`	the arithmetic mean of precision (proportion of predicted presences actually observed as presences) across all threshold values (defined by 'interval'). It is close to the AUC of the precision-recall (PR) curve.
`GiniCoefficient`	the Gini coefficient (aka Gini index or Gini ratio), a measure of statistical dispersion intended to represent inequality, computed as 2*AUC-1.
`SomersD`	Somers' D, a measure of ordinal association between two variables (i.e. a rank correlation), also used as a quality measure of binary choice, computed as 2*(AUC-0.5). It is mathematically equivalent to the Gini coefficient; they are both here to match the terminology familiar to different audiences.

Author(s)

A. Marcia Barbosa

References

Flach, P. & Kull, M. (2015) Precision-recall-gain curves: PR analysis done right. Pages 838-846 in C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors. Advances in neural information processing systems. Volume 1. Massachusetts Institute of Technology (MIT) Press, Red Hook, New York, USA

Lobo, J.M., Jimenez-Valverde, A. & Real, R. (2008) AUC: a misleading measure of the performance of predictive distribution models. Global Ecology and Biogeography 17: 145-151

Jimenez-Valverde, A., Acevedo, P., Barbosa, A.M., Lobo, J.M. & Real, R. (2013) Discrimination capacity in species distribution models depends on the representativeness of the environmental domain. Global Ecology and Biogeography 22: 508-516

Sofaer, H.R., Hoeting, J.A. & Jarnevich, C.S. (2019). The area under the precision-recall curve as a performance metric for rare binary events. Methods in Ecology and Evolution, 10: 565-577

Swets, J.A. (1988) Measuring the accuracy of diagnostic systems. Science, 240: 1285-1293 (doi: 10.1126/science.3287615)

See Also

threshMeasures

Examples

```
# load sample models:
data(rotif.mods)

# choose a particular model to play with:
mod <- rotif.mods$models[[1]]


# compute the AUC:

AUC(model = mod)

AUC(model = mod, simplif = TRUE)

AUC(model = mod, curve = "PR")

AUC(model = mod, interval = 0.1, grid = TRUE)

AUC(model = mod, plot.preds = TRUE)

AUC(model = mod, ticks = TRUE)

AUC(model = mod, plot.preds = c("curve", "bottom"))


# you can also use vectors of observed and predicted values
# instead of a model object:

presabs <- mod$y
```

```
prediction <- mod$fitted.values

AUC(obs = presabs, pred = prediction)


# 'obs' can also be a table of presence point coordinates
# and 'pred' a SpatRaster of predicted values
```

Boyce *Boyce Index*

Description

This function computes the (continuous) Boyce index (Boyce 2002; Hirzel et al. 2006) for either: 1) a model object; or 2) two paired numeric vectors of observed (binary, 1 for occurrence vs. 0 for no occurrence records) and predicted (continuous, e.g. environmental suitability) values; or 3) a set of presence point coordinates and a raster map with the predicted values for the entire model evaluation area.

This metric is designed for evaluating model predictions against presence/background data (i.e. presence/available, where "available" includes both occupied and unoccupied sites; Boyce 2002), so the function uses the model predictions for the presence sites (ones) against the predictions for the entire dataset (ones and zeros).

Usage

```
Boyce(model = NULL, obs = NULL, pred = NULL, n.bins = NA,
bin.width = "default", res = 100, method = "spearman", rm.dup.classes = FALSE,
rm.dup.points = FALSE, pbg = FALSE, simplif = FALSE, plot = TRUE,
plot.lines = TRUE, plot.values = TRUE, plot.digits = 3,
xlim = c(0, 1), ylim = "auto", na.rm = TRUE, verbosity = 2, ...)
```

Arguments

model	optional binary-response model object of class "glm", "gam", "gbm", "randomForest" or "bart". If this argument is provided, 'obs' and 'pred' will be extracted with mod2obspred. Alternatively, you can input the 'obs' and 'pred' arguments (e.g. for external test data) instead of 'model'.
obs	alternatively to 'model' and together with 'pred', a numeric vector of observed presences (1) and absences (0) of a binary response variable. Alternatively (and if 'pred' is a 'SpatRaster'), a two-column matrix or data frame containing, respectively, the x (longitude) and y (latitude) coordinates of the presence points, in which case the 'obs' vector will be extracted with ptsrast2obspred. This argument is ignored if 'model' is provided.
pred	alternatively to 'model' and together with 'obs', a vector with the corresponding predicted values of presence probability, habitat suitability, environmental favourability or alike. Must be of the same length and in the same order as 'obs'. Alternatively (and if 'obs' is a set of point coordinates), a 'SpatRaster' map of

	the predicted values for the entire evaluation region, in which case the 'pred' vector will be extracted with ptsrast2obspred. This argument is ignored if 'model' is provided.
n.bins	number of classes or bins (e.g. 10) in which to group the 'pred' values, or a vector with the bin thresholds. If n.bins = NA (the default), a moving window is used (see next parameters), so as to compute the "continuous Boyce index" (Hirzel et al. 2006).
bin.width	width of the moving window (if n.bins = NA), in the units of 'pred' (e.g. 0.1). By default, it is 1/10th of the 'pred' range.
res	resolution of the moving window (if n.bins = NA). By default it is 100 focals, providing 100 moving bins).
method	argument to be passed to cor indicating which correlation coefficient to use. The default is 'spearman' as per Boyce et al. (2002), but 'pearson' and 'kendall' could be more informative in many cases.
rm.dup.classes	Logical, default FALSE. If TRUE (as in 'ecospat::ecospat.boyce') and if there are different bins with the same predicted/expected ratio, only one of each is used to compute the correlation (see Examples).
rm.dup.points	Logical (default FALSE); if TRUE and if 'pred' is a SpatRaster and if there are >1 points within the same pixel, a maximum of one point per pixel is used to compute the presences. See examples in ptsrast2obspred.
pbg	logical (default FALSE) to pass to inputMunch indicating whether to use presence/background (rather than presence/absence) data.
simplif	logical value (default FALSE) indicating whether to return just a simplified output containing only the value estimate, with no associated statistics.
plot	logical (default TRUE), whether or not to plot the predicted/expected ratio against the median prediction of each bin.
plot.lines	logical (default TRUE), whether or not to add lines connecting the points in the plot (if plot=TRUE).
plot.values	logical (default TRUE), whether or not to show in the plot the value of the Boyce index.
plot.digits	number of digits to which the value in the plot should be rounded (if 'plot' and 'plot.values' are TRUE). Defaults to 3.
xlim	either "auto" for fitting the x axis to the existing values, or a length-2 numeric vector setting its limits. The default is c(0, 1).
ylim	either "auto" (default) for fitting the y axis to the existing values, or a length-2 numeric vector setting its limits (e.g. to compare Boyce plots across different datasets).
na.rm	Logical (default TRUE) value indicating if missing values should be removed from computations.
verbosity	integer specifying the amount of messages to display. Defaults to the maximum implemented; lower numbers (down to 0) decrease the number of messages.
...	some additional arguments can be passed to plot, e.g. 'main' or 'xlim'.

Details

The Boyce index is the correlation between model predictions and area-adjusted frequencies (i.e., observed vs. expected proportion of occurrences) along different prediction classes (bins). In other words, it measures how model predictions differ from a random distribution of the observed presences across the prediction gradient (Boyce et al. 2002). It can take values between -1 and 1. Positive values indicate that presences are more frequent than expected by chance (given availability) in areas with higher predicted values. Values close to zero mean that predictions are no better than random (i.e. presences are distributed among prediction classes as expected by chance), and negative values indicate counter predictions (i.e., presences are proportionally more frequent in areas with lower predicted values).

The R code is largely based on the 'ecospat.boyce' function of the **ecospat** package (version 3.2.1), but it is modified to match the input formats in the remaining functions of 'modEvA', and to return a more complete and informative output. It also does not 'rm.dup.classes' by default, like the 'ecospat' version does (though the user can control this parameter); and it implements a bug fix identified at https://github.com/ecospat/ecospat/issues/99#issue-2528136160.

Value

This function returns a `list` with the following components:

`bins`	a data frame with the number of values in each bin, their median and range of predicted values, and the corresponding predicted/expected ratio of presences.
`B`	the numeric value of the Boyce index, i.e. the coefficient of correlation between the median predicted value in each bin and the corresponding predicted/expected ratio.

If plot=TRUE (the default), the function also plots the predicted/expected ratio for the utilized bins along the prediction range. A good model should yield a monotonically increasing curve (but see Note).

Note

This index is designed for evaluating predictions of habitat suitability, not presence probability (which also depends on the species' presence/absence ratio: rare species do not usually show high proportions of presences, even in highly suitable areas). If your predictions are of presence probability based on a training prevalence different from 50% presences, it is generally better to convert those predictions e.g. with the `Fav` function of package **fuzzySim**, before evaluating them with the Boyce index.

In bins with overly small sample sizes, the comparison between median prediction and random expectation may not be meaningful, although these bins will equally contribute to the overall Boyce index. When there are bins with fewer than 15 values (the minimum required for reliable comparisons within bins; Jovani & Tella 2006, Jimenez-Valverde et al. 2013), a warning is emitted and their values are plotted as red triangles. See the $bins$bin.N section of the console output, and use the 'bin.width' argument to enlarge the bins if necessary.

Author(s)

A. Marcia Barbosa, with significant chunks of code from the 'ecospat::ecospat.boyce' function by Blaise Petitpierre and Frank Breiner (**ecospat** package version 3.2.1).

References

Boyce, M.S., P.R. Vernier, S.E. Nielsen & F.K.A. Schmiegelow (2002) Evaluating resource selection functions. Ecological Modelling 157: 281-300

Hirzel, A.H., G. Le Lay, V. Helfer, C. Randin & A. Guisan (2006) Evaluating the ability of habitat suitability models to predict species presences. Ecological Modelling 199: 142-152

Jimenez-Valverde A., Acevedo P., Barbosa A.M., Lobo J.M. & Real R. (2013) Discrimination capacity in species distribution models depends on the representativeness of the environmental domain. Global Ecology and Biogeography 22: 508-516

Jovani R. & Tella J.L. (2006) Parasite prevalence and sample size: misconceptions and solutions. Trends in Parasitology 22: 214-218

See Also

Miller; HLfit; ecospat::ecospat.boyce; enmSdmX::evalContBoyce

Examples

```
# load sample models:
data(rotif.mods)

# choose a particular model to play with:
mod <- rotif.mods$models[[1]]

# compute the Boyce index:
Boyce(model = mod, main = "My model Boyce plot")
Boyce(model = mod, main = "My model Boyce plot", rm.dup.classes = TRUE)


# you can also use vectors of observed and predicted values
# instead of a model object:

presabs <- mod$y
prediction <- mod$fitted.values

Boyce(obs = presabs, pred = prediction)


# 'obs' can also be a table of presence point coordinates
# and 'pred' a SpatRaster of predicted values
```

---

confusionLabel *Label predictions according to their confusion matrix category*

Description

This function labels the (typically continuous) predictions of a binary-response model according to their confusion matrix categories, i.e., it classifies each prediction into a false positive, false negative, true positive or true negative, given a user-defined threshold value.

Usage

```
confusionLabel(model = NULL, obs = NULL, pred = NULL, thresh,
interval = 0.01, quant = 0, verbosity = 2, na.rm = FALSE,
rm.dup = FALSE, plot = TRUE, ...)
```

Arguments

model	optional binary-response model object of class "glm", "gam", "gbm", "randomForest" or "bart". If this argument is provided, 'obs' and 'pred' will be extracted with mod2obspred. Alternatively, you can input the 'obs' and 'pred' arguments (e.g. for external test data) instead of 'model'.
obs	alternatively to 'model' and together with 'pred', a numeric vector of observed presences (1) and absences (0) of a binary response variable. Alternatively (and if 'pred' is a 'SpatRaster'), a two-column matrix or data frame containing, respectively, the x (longitude) and y (latitude) coordinates of the presence points, in which case the 'obs' vector will be extracted with ptsrast2obspred. This argument is ignored if 'model' is provided.
pred	alternatively to 'model' and together with 'obs', a vector with the corresponding predicted values of presence probability, habitat suitability, environmental favourability or alike. Must be of the same length and in the same order as 'obs'. Alternatively (and if 'obs' is a set of point coordinates), a 'SpatRaster' map of the predicted values for the entire evaluation region, in which case the 'pred' vector will be extracted with ptsrast2obspred. This argument is ignored if 'model' is provided.
thresh	numeric value of the threshold to separate predicted presences from predicted absences; can be "preval", to use the prevalence of 'obs' (or of the response variable in 'model') as the threshold, or any real number between 0 and 1. See Details in threshMeasures for an informed choice.
interval	numeric value, used if 'thresh' is a threshold optimization method such as "maxKappa" or "maxTSS", indicating the interval between the thresholds to test. The default is 0.01. Smaller values may provide more precise results but take longer to compute.
quant	numeric value indicating the proportion of presences to discard if thresh="MTP" (minimum training presence). With the default value 0, MTP will be the threshold at which all observed presences are classified as such; with e.g. quant=0.05, MTP will be the threshold at which 5% presences will be classified as absences.
verbosity	integer specifying the amount of messages to display. Defaults to the maximum implemented; lower numbers (down to 0) decrease the number of messages.
na.rm	logical argument indicating whether to remove (with a warning saying how many) rows with NA in any of the 'obs' or 'pred' values. The default is FALSE.
rm.dup	if TRUE and if 'pred' is a SpatRaster and if there are repeated points within the same pixel, a maximum of one point per pixel is used to compute the presences. See examples in ptsrast2obspred. The default is FALSE.
plot	if TRUE (the default) and if 'pred' is a SpatRaster, the output (also a SpatRaster) is automatically plotted. Map categories have a set colour table, built with terra::coltab().

...	additional arguments to pass to terra::plot() (if 'pred' is a SpatRaster and plot=TRUE), such as 'mar', 'axes' or 'legend'.

Value

This function returns a categorical (factor) vector (or a categorical 'SpatRaster' if 'pred' is of that class) of the same length as 'pred', or of the same number of rows as the data in 'model', containing the confusion matrix label for each value.

Author(s)

A. Marcia Barbosa

See Also

threshMeasures, confusionMatrix

Examples

```
# load sample models:
data(rotif.mods)

# choose a particular model to play with:
mod <- rotif.mods$models[[1]]

confusionLabel(model = mod, thresh = 0.5)


# you can instead use vectors of observed and predicted values:

presabs <- mod$y
prediction <- mod$fitted.values

confusionLabel(obs = presabs, pred = prediction, thresh = 0.5)


# 'obs' can also be a table of presence point coordinates
# and 'pred' a SpatRaster of predicted values
```

---

confusionMatrix *Confusion matrix*

Description

This function computes the confusion (or contingency) matrix for a binary-response model, containing the numbers of false positives, false negatives, true positives and true negatives, given a user-defined threshold value.

Usage

```
confusionMatrix(model = NULL, obs = NULL, pred = NULL, thresh, interval = 0.01,
quant = 0, verbosity = 2, na.rm = TRUE, rm.dup = FALSE, plot = FALSE,
classes = FALSE, ...)
```

Arguments

model	optional binary-response model object of class "glm", "gam", "gbm", "randomForest" or "bart". If this argument is provided, 'obs' and 'pred' will be extracted with mod2obspred. Alternatively, you can input the 'obs' and 'pred' arguments (e.g. for external test data) instead of 'model'.
obs	alternatively to 'model' and together with 'pred', a numeric vector of observed presences (1) and absences (0) of a binary response variable. Alternatively (and if 'pred' is a 'SpatRaster'), a two-column matrix or data frame containing, respectively, the x (longitude) and y (latitude) coordinates of the presence points, in which case the 'obs' vector will be extracted with ptsrast2obspred. This argument is ignored if 'model' is provided.
pred	alternatively to 'model' and together with 'obs', a vector with the corresponding predicted values of presence probability, habitat suitability, environmental favourability or alike. Must be of the same length and in the same order as 'obs'. Alternatively (and if 'obs' is a set of point coordinates), a 'SpatRaster' map of the predicted values for the entire evaluation region, in which case the 'pred' vector will be extracted with ptsrast2obspred. This argument is ignored if 'model' is provided.
thresh	numeric value of the threshold to separate predicted presences from predicted absences; can be "preval", to use the prevalence of 'obs' (or of the response variable in 'model') as the threshold, or any real number between 0 and 1. See Details in threshMeasures for an informed choice.
interval	numeric value, used if 'thresh' is a threshold optimization method such as "maxKappa" or "maxTSS", indicating the interval between the thresholds to test. The default is 0.01. Smaller values may provide more precise results but take longer to compute.
quant	numeric value indicating the proportion of presences to discard if thresh="MTP" (minimum training presence). With the default value 0, MTP will be the threshold at which all observed presences are classified as such; with e.g. quant=0.05, MTP will be the threshold at which 5% presences will be classified as absences.
verbosity	integer specifying the amount of messages to display. Defaults to the maximum implemented; lower numbers (down to 0) decrease the number of messages.
na.rm	logical argument indicating whether to remove (with a warning saying how many) rows with NA in any of the 'obs' or 'pred' values. The default is FALSE.
rm.dup	if TRUE and if 'pred' is a SpatRaster and if there are repeated points within the same pixel, a maximum of one point per pixel is used to compute the presences. See examples in ptsrast2obspred. The default is FALSE.
plot	logical argument indicating whether to also plot the matrix as an image. The default is FALSE (for back-compatibility).

classes	logical argument indicating whether the matrix image (if plot=TRUE) should have qualitative colours, matching the output of confusionLabel for SpatRasters. The default is FALSE, in which case the colours are proportional to the values in each section of the matrix, and the palette can be user-specified with the 'col' argument for 'plot' (see Examples).
...	some additional arguments can be passed to image (and through to plot) if plot=TRUE, such as 'main', 'font.main' or 'cex.main' (not 'axes', 'xlab' or 'ylab', which are already defined by confusionMatrix).

Value

This function returns a data frame containing the four values of the confusion matrix.

Author(s)

A. Marcia Barbosa

See Also

threshMeasures, confusionLabel

Examples

```
# load sample models:
data(rotif.mods)

# choose a particular model to play with:
mod <- rotif.mods$models[[1]]

confusionMatrix(model = mod, thresh = 0.5)

confusionMatrix(model = mod, thresh = 0.5, plot = TRUE)

confusionMatrix(model = mod, thresh = 0.5, plot = TRUE,
col = hcl.colors(100, "blues"))

confusionMatrix(model = mod, thresh = 0.5, plot = TRUE, classes = TRUE,
main = "Confusion matrix")


# you can instead use vectors of observed and predicted values:

presabs <- mod$y
prediction <- mod$fitted.values

confusionMatrix(obs = presabs, pred = prediction, thresh = 0.5, plot = TRUE)


# 'obs' can also be a table of presence point coordinates
# and 'pred' a SpatRaster of predicted values
```

`Dsquared` *Explained deviance*

Description

This function computes the (adjusted) amount of deviance accounted for by a model, given a model object or a set of observed and predicted values.

Usage

```
Dsquared(model = NULL, obs = NULL, pred = NULL, family = NULL,
adjust = FALSE, npar = NULL, na.rm = TRUE, rm.dup = FALSE, pbg = FALSE,
dismo.version = FALSE, verbosity = 2)
```

Arguments

`model`	optional model object of a class implemented in `mod2obspred`. If this argument is provided, 'obs' and 'pred' will be extracted with that function. Alternatively, you can input the 'obs' and 'pred' arguments instead of 'model'.
`obs`	alternatively to 'model' and together with 'pred', a numeric vector of observed values of the response variable. Alternatively (and if 'pred' is a 'SpatRaster'), a two-column matrix or data frame containing, respectively, the x (longitude) and y (latitude) coordinates of presence points, in which case the 'obs' vector of presences and absences will be extracted with `ptsrast2obspred`. This argument is ignored if 'model' is provided.
`pred`	alternatively to 'model' and together with 'obs', a vector with the corresponding predicted values, of the same length and in the same order as 'obs'. Alternatively (and if 'obs' is a set of point coordinates), a 'SpatRaster' map of the predicted values for the entire evaluation region, in which case the 'pred' vector will be extracted with `ptsrast2obspred`. This argument is ignored if 'model' is provided.
`family`	a character vector (i.e. in quotes) of length 1 specifying the family of the model that generated the 'pred' values. This argument is ignored if `model` is provided and is of a class for which `family` provides a result; otherwise (i.e. if 'obs' and 'pred' are provided rather than a model object), family can be specified by the user, or (if left NULL) will be guessed (with a message) given the values of the response variable.
`adjust`	logical, whether or not to adjust the D-squared value for the number of observations and parameters in the model (see Details). The default is `FALSE`; `TRUE` requires either providing the `model` object of class GLM, or specifying the number of parameters in the model that produced the `pred` values.
`npar`	integer value indicating the number of parameters in the model. This argument is ignored and taken from `model` if this argument is provided and of class GLM, or if `adjust = FALSE`.
`na.rm`	Logical value indicating whether missing values should be ignored in computations. Defaults to TRUE.

rm.dup	If TRUE and if 'pred' is a SpatRaster and if there are repeated points within the same pixel, a maximum of one point per pixel is used to compute the presences. See examples in ptsrast2obspred. The default is FALSE.
pbg	logical value to pass to inputMunch indicating whether to use presence/background (rather than presence/absence) data. Default FALSE.
dismo.version	Logical value indicating whether the deviance should be computed with code from the dismo::calc.deviance() function. The default is FALSE, for back-compatibility.
verbosity	integer specifying the amount of messages to display. Defaults to the maximum implemented; lower numbers (down to 0) decrease the number of messages.

Details

Linear models have an R-squared value (commonly provided with the model summary) which measures the proportion of variation that the model accounts for. For generalized linear models (GLMs) and others based on non-continuous response variables, an equivalent is the amount of deviance accounted for (D-squared; Guisan & Zimmermann 2000), though this value is not routinely provided with the model summary. The Dsquared function calculates it as the proportion of the null deviance (i.e. the deviance of a model with no predictor variables) that is accounted for by the model. There is also an option to compute the adjusted D-squared, which takes into account the number of observations and the number of parameters, thus allowing direct comparison among the output for different models (Weisberg 1980, Guisan & Zimmermann 2000).

The function computes the mean residual deviance (as in the calc.deviance function of package **dismo**) of the observed (response) against the predicted values, and the mean deviance of a null model (with no predictor variables), i.e. of the response against the mean of the response. Finally, it gets the explained deviance as (null-residual)/null.

Value

The function returns a numeric value indicating the (optionally adjusted) proportion of deviance accounted for by the input model predictions.

Author(s)

A. Marcia Barbosa, with parts of code from 'dismo::calc.deviance' by John R. Leathwick and Jane Elith

References

Guisan, A. & Zimmermann, N.E. (2000) Predictive habitat distribution models in ecology. Ecological Modelling 135: 147-186

Weisberg, S. (1980) Applied Linear Regression. Wiley, New York

See Also

plotGLM, pseudoRsq, dismo::calc.deviance

Examples

```
# load sample models:
data(rotif.mods)

# choose a particular model to play with:
mod <- rotif.mods$models[[1]]

Dsquared(model = mod)

Dsquared(model = mod, adjust = TRUE)


# you can also use Dsquared with vectors of observed and predicted values
# instead of with a model object:

presabs <- mod$y
prediction <- mod$fitted.values
parameters <- attributes(logLik(mod))$df

Dsquared(obs = presabs, pred = prediction, family = "binomial")

Dsquared(obs = presabs, pred = prediction, family = "binomial",
adjust = TRUE, npar = parameters)


# 'obs' can also be a table of presence point coordinates
# and 'pred' a SpatRaster of predicted values
```

`errorMeasures` *Measures of model prediction error.*

Description

This function computes metrics such as the (root) mean square error or the Brier score, of a model object or a set of observed and predicted values or maps. The smaller the returned values, the better the modol predictions fit the observations.

Usage

```
errorMeasures(model = NULL, obs = NULL, pred = NULL, na.rm = TRUE,
rm.dup = FALSE, verbosity = 2)
```

Arguments

`model`	optional model object of class implemented in `mod2obspred`. If this argument is provided, 'obs' and 'pred' will be extracted with that function. Alternatively, you can input the 'obs' and 'pred' arguments instead of 'model'.

obs	alternatively to 'model' and together with 'pred', a numeric vector of observed values of the response variable. Alternatively (and if 'pred' is a 'SpatRaster'), a two-column matrix or data frame containing, respectively, the x (longitude) and y (latitude) coordinates of presence points, in which case the 'obs' vector of presences and absences will be extracted with ptsrast2obspred. This argument is ignored if 'model' is provided.
pred	alternatively to 'model' and together with 'obs', a vector with the corresponding predicted values, of the same length and in the same order as 'obs'. Alternatively (and if 'obs' is a set of point coordinates), a 'SpatRaster' map of the predicted values for the entire evaluation region, in which case the 'pred' vector will be extracted with ptsrast2obspred. This argument is ignored if 'model' is provided.
na.rm	Logical value indicating whether missing values should be ignored in computations. Defaults to TRUE.
rm.dup	If TRUE and if 'pred' is a SpatRaster and if there are repeated points within the same pixel, a maximum of one point per pixel is used to compute the presences. See examples in ptsrast2obspred. The default is FALSE.
verbosity	integer specifying the amount of messages to display. Defaults to the maximum implemented; lower numbers (down to 0) decrease the number of messages.

Details

The Brier score measures the accuracy of probabilistic predictions against a binary response variable. It is computed as the mean of the squared differences between observed and predicted values, i.e. the mean square error (MSE). It may take values between 0 and 1. The smaller the score, the better the model fit.

The root mean square error is more usually applied to models of coninuous response variables. It is computed as the square root of the mean of the squared differences between observed and predicted values. It is (approximately) the same as the standard deviation of the model residuals (prediction errors), i.e., a measure of how spread out these residuals are, or how concentrated the observations are around the model prediction line.

Value

This function returns a named list of numeric values indicating various measures of model error.

Author(s)

A. Marcia Barbosa

References

Kenney J.F. & Keeping E.S. (1962) Root Mean Square. "Mathematics of Statistics", 3rd ed. Princeton, NJ: Van Nostrand, pp. 59-60.

See Also

RMSE, plotGLM, pseudoRsq, Dsquared

Examples

```
# load sample models:
data(rotif.mods)

# choose a particular model to play with:
mod <- rotif.mods$models[[1]]

errorMeasures(model = mod)


# you can also use errorMeasures with vectors of observed and predicted values
# instead of with a model object:

presabs <- mod$y
prediction <- mod$fitted.values

errorMeasures(obs = presabs, pred = prediction)


# 'obs' can also be a table of presence point coordinates
# and 'pred' a SpatRaster of predicted values
```

`evaluate` *Evaluate a model based on the elements of a confusion matrix.*

Description

This function evaluates the classification performance of a model based on the values of a confusion matrix obtained at a particular threshold.

Usage

```
evaluate(a, b, c, d, N = NULL, measure = "CCR")
```

Arguments

`a`	number of correctly predicted presences (true positives)
`b`	number of absences incorrectly predicted as presences (false positives)
`c`	number of presences incorrectly predicted as absences (false negatives)
`d`	number of correctly predicted absences (true negatives)
`N`	total number of cases. If NULL (the default), it is calculated automatically by adding up a, b, c and d.)
`measure`	a character vector of length 1 indicating the evaluation measure to use. Type `modEvAmethods("threshMeasures")` for available options.

Details

A number of measures can be used to evaluate continuous model predictions against observed binary occurrence data (Fielding & Bell 1997; Liu et al. 2011; Barbosa et al. 2013; Leroy et al. 2018). The 'evaluate' function can calculate a few threshold-based classification measures from the values of a confusion matrix obtained at a particular threshold. The 'evaluate' function is used internally by threshMeasures. It can also be accessed directly by the user, but it is usually more practical to use 'threshMeasures', which calculates the confusion matrix automatically.

Value

Numeric value of the specified evaluation measure.

Note

Some measures (e.g. NMI, odds ratio) don't work with zeros in (some parts of) the confusion matrix. Also, TSS and NMI are not symmetrical, i.e. "obs" vs "pred" is different from "pred" vs "obs".

Author(s)

A. Marcia Barbosa

References

Barbosa A.M., Real R., Munoz A.R. & Brown J.A. (2013) New measures for assessing model equilibrium and prediction mismatch in species distribution models. Diversity and Distributions, 19: 1333-1338

Fielding A.H. & Bell J.F. (1997) A review of methods for the assessment of prediction errors in conservation presence/absence models. Environmental Conservation 24: 38-49

Leroy B., Delsol R., Hugueny B., Meynard C.M., Barhoumi C., Barbet-Massin M. & Bellard C. (2018) Without quality presence-absence data, discrimination metrics such as TSS can be misleading measures of model performance. Journal of Biogeography 45(9):1994-2002

Liu C., White M., & Newell G. (2011) Measuring and comparing the accuracy of species distribution models with presence-absence data. Ecography, 34, 232-243.

See Also

threshMeasures

Examples

```
evaluate(23, 44, 21, 34)

evaluate(23, 44, 21, 34, measure = "TSS")
```

evenness *Evenness in a binary vector.*

Description

For building and evaluating species distribution models, the porportion of presences (prevalence) of a species and the balance between the number of presences and absences may be issues to take into account (e.g. Jimenez-Valverde & Lobo 2006, Barbosa et al. 2013). The evenness function calculates the presence-absence balance in a binary (e.g., presence/absence) vector.

Usage

```
evenness(obs)
```

Arguments

obs	a vector of binary observations (e.g. 1 or 0, male or female, disease or no disease, etc.)

Value

A number ranging between 0 when all values are the same, and 1 when there are the same number of cases with each value in obs.

Author(s)

A. Marcia Barbosa

References

Barbosa A.M., Real R., Munoz A.R. & Brown J.A. (2013) New measures for assessing model equilibrium and prediction mismatch in species distribution models. Diversity and Distributions, 19: 1333-1338

Jimenez-Valverde A. & Lobo J.M. (2006) The ghost of unbalanced species distribution data in geographical model predictions. Diversity and Distributions, 12: 521-524.

See Also

prevalence

Examples

```
(x <- rep(c(0, 1), each = 5))
(y <- c(rep(0, 3), rep(1, 7)))
(z <- c(rep(0, 7), rep(1, 3)))

prevalence(x)
evenness(x)
```

```
prevalence(y)
evenness(y)

prevalence(z)
evenness(z)
```

`getBins` *Get bins of continuous values.*

Description

Get continuous predicted values into bins according to specific criteria.

Usage

```
getBins(model = NULL, obs = NULL, pred = NULL, id = NULL,
bin.method, n.bins = ifelse(bin.method != "mov.bins", 10, 100),
fixed.bin.size = FALSE, min.bin.size = 15,
min.prob.interval = 0.1, bin.width = "default", quantile.type = 7,
simplif = FALSE, verbosity = 2, na.rm = TRUE, rm.dup = FALSE)
```

Arguments

`model`	optional binary-response model object of class "glm", "gam", "gbm", "randomForest" or "bart". If this argument is provided, 'obs' and 'pred' will be extracted with `mod2obspred`. Alternatively, you can input the 'obs' and 'pred' arguments instead of 'model'.
`obs`	alternatively to 'model' and together with 'pred', a numeric vector of observed presences (1) and absences (0) of a binary response variable. Alternatively (and if 'pred' is a 'SpatRaster'), a two-column matrix or data frame containing, respectively, the x (longitude) and y (latitude) coordinates of the presence points, in which case the 'obs' vector will be extracted with `ptsrast2obspred`. This argument is ignored if 'model' is provided.
`pred`	alternatively to 'model' and together with 'obs', a vector with the corresponding predicted values of presence probability, habitat suitability, environmental favourability or alike. Must be of the same length and in the same order as 'obs'. Alternatively (and if 'obs' is a set of point coordinates), a 'SpatRaster' map of the predicted values for the entire evaluation region, in which case the 'pred' vector will be extracted with `ptsrast2obspred`. This argument is ignored if 'model' is provided.
`id`	optional vector of row identifiers; must be of the same length and in the same order of `obs` and `pred` (or of the cases used to build `model`)
`bin.method`	the method with which to divide the values into bins. Type modEvAmethods("getBins") for available options and see Details for more information on these methods.

n.bins	the number of bins in which to divide the data.
fixed.bin.size	logical, whether all bins should have (approximately) the same size.
min.bin.size	integer value defining the minimum number of observations to include in each bin. The default is 15, the minimum required for accurate comparisons within bins (Jovani & Tella 2006, Jimenez-Valverde et al. 2013).
bin.width	width of the moving window (if method = "mov.bins"), in the units of 'pred' (e.g. 0.1). By default, it is 1/10th of the 'pred' range.
min.prob.interval	minimum range of prredicted values in each bin (default 0.1).
quantile.type	argument to pass to quantile specifying the algorithm to use if bin.method = "quantiles". The default is 7 (the quantile default in R), but check out other types, e.g. 3 (used by SAS), 6 (used by Minitab and SPSS) or 5 (appropriate for deciles, which correspond to the default n.bins = 10).
simplif	logical (default FALSE), whether to calculate a faster, simplified version (used internally in other functions).
verbosity	integer specifying the amount of console messages or warnings to display. Defaults to the maximum implemented; lower numbers (down to 0) decrease the number of messages.
na.rm	logical, whether to remove (with a warning saying how many) rows with NA in any of the 'obs' or 'pred' values. The default is TRUE, as some 'bin.method' options will fail if there are NAs.
rm.dup	logical (default FALSE). If TRUE and 'pred' is a SpatRaster and there are >1 points in the same pixel, a maximum of 1 point per pixel is used to compute the presences (see examples in ptsrast2obspred).

Details

Mind that different bin.methods can lead to visibly different results regarding the bins and any operations that depend on them (such as HLfit and Boyce). Currently available bin.methods are:

- round.prob: predicted values are rounded to the number of digits of min.prob.interval - e.g., if min.prob.interval = 0.1 (the default), values under 0.05 get into bin 1 (rounded probability = 0), values between 0.05 and 0.15 get into bin 2 (rounded probability = 0.1), etc. until values with probability over 0.95, which get into bin 11. Arguments n.bins, fixed.bin.size and min.bin.size are ignored by this bin.method.

- prob.bins: predicted values are grouped into bins of the given intervals - e.g., if min.prob.interval = 0.1 (the default), bin 1 gets the values between 0 and 0.1, bin 2 gets the values between 0.1 and 0.2, etc. until bin 10, which gets the values between 0.9 and 1. Arguments n.bins, fixed.bin.size and min.bin.size are ignored by this bin.method.

- size.bins: predicted values are grouped into bins of (approximately) equal size, defined by argument min.bin.size. Arguments n.bins and min.prob.interval are ignored by this bin.method.

- n.bins: predicted values are divided into the number of bins given by argument n.bins, and their sizes may or may not be forced to be (approximately) equal, depending on argument fixed.bin.size (which is FALSE by default). Arguments min.bin.size and min.prob.interval are ignored by this bin.method.

- quantiles: predicted values are divided using R function quantile, with cutpoints defined by the given n.bins (i.e., deciles by default), and with the quantile algorithm defined by argument quantile.type. Arguments fixed.bin.size, min.bin.size and min.prob.interval are ignored by this bin.method.

- mov.bins: predicted values are grouped into moving bins (i.e., bins defined by a moving window), which partially overlap, with cutpoints defined by bin.width. Arguments fixed.bin.size, min.bin.size and min.prob.interval are ignored by this bin.method.

Value

The output of getBins is a list with the following components:

prob.bin	the bin in which each observation falls.
bins.table	a data frame with the sample size, number of presences, number of absences, prevalence, mean and median predicted value, and the difference between predicted and observed values (mean predicted value minus observed prevalence) in each bin.
N	the total number of observations in the analysis.
n.bins	the total number of bins obtained.

Note

This function is still under development and may fail for some datasets and binning methods (e.g., ties may sometimes preclude binning under some bin.methods). Fixes and other improvements are under way. Feedback welcome!

References

Jimenez-Valverde A., Acevedo P., Barbosa A.M., Lobo J.M. & Real R. (2013) Discrimination capacity in species distribution models depends on the representativeness of the environmental domain. Global Ecology and Biogeography 22: 508-516

Jovani R. & Tella J.L. (2006) Parasite prevalence and sample size: misconceptions and solutions. Trends in Parasitology 22: 214-218

See Also

HLfit

Examples

```
# load sample models:

data(rotif.mods)


# choose a particular model to play with:

mod <- rotif.mods$models[[1]]
```

```
# try getBins using different binning methods:

getBins(model = mod, bin.method = "quantiles")

getBins(model = mod, bin.method = "n.bins")

getBins(model = mod, bin.method = "n.bins", fixed.bin.size = TRUE)
```

---

getModEqn *Get model equation*

Description

This function retrieves the regression equation of a linear or generalized linear model, to print or apply elsewhere.

Usage

```
getModEqn(model, type = "Y", digits = NULL, prefix = NULL,
suffix = NULL)
```

Arguments

model	a model object of class 'lm' or glm'.
type	the type of equation to get; can be either "Y" (the default, for the linear model equation), "P" (for probabiity) or "F" (for favourability).
digits	the number of significant digits to which to round the coefficient estimates in the equation.
prefix	the prefix to add to each variable name in the equation.
suffix	the suffix to add to each variable name in the equation.

Details

The summary of a model in R gives you a table of the coefficient estimates and other parameters. Sometimes it may be useful to have a string of text with the model's equation, so that you can present it in an article (e.g. Real et al. 2005) or apply it in a (raster map) calculation, either in R (although here you can normally use the 'predict' function for this) or in another GIS software (e.g. Barbosa et al. 2010). The getModEqn function gets this equation for linear or generalized linear models.

By default it prints the "Y" linear equation, but for generalized linear models you can also set type = "P" (for the equation of probability) or type = "F" (for favourability, which modifies the intercept to eliminate the effect of modelled prevalence - see Real et al. 2006).

If the variables to which you want to apply the model have a prefix or suffix (e.g. something like prefix = "raster.stack$" for the R 'raster' or 'terra' package, or prefix = "mydata$" for a data frame, or suffix = "@1" in QGIS, or suffix = "@mapset" in GRASS), you can get these in the equation too, using the prefix and/or the suffix argument.

Value

A charachter string of the model equation.

Author(s)

A. Marcia Barbosa

References

Barbosa A.M., Real R. & Vargas J.M. (2010) Use of coarse-resolution models of species' distributions to guide local conservation inferences. Conservation Biology 24: 1378-87

Real R., Barbosa A.M., Martinez-Solano I. & Garcia-Paris, M. (2005) Distinguishing the distributions of two cryptic frogs (Anura: Discoglossidae) using molecular data and environmental modeling. Canadian Journal of Zoology 83: 536-545

Real R., Barbosa A.M. & Vargas J.M. (2006) Obtaining environmental favourability functions from logistic regression. Environmental and Ecological Statistics 13: 237-245

Examples

```
# load sample models:
data(rotif.mods)

# choose a particular model to play with:
mod <- rotif.mods$models[[1]]

getModEqn(mod)

getModEqn(mod, type = "P", digits = 3, suffix = "@mapset")

getModEqn(mod, type = "F", digits = 2)
```

---

getThreshold *Prediction threshold for a given criterion*

Description

This function computes the prediction threshold under a given criterion.

Usage

```
getThreshold(model = NULL, obs = NULL, pred = NULL, threshMethod,
interval = 0.01, quant = 0, na.rm = TRUE, verbosity = 2, pbg = FALSE)
```

Arguments

model	optional binary-response model object of class "glm", "gam", "gbm", "randomForest" or "bart". If this argument is provided, 'obs' and 'pred' will be extracted with mod2obspred. Alternatively, you can input the 'obs' and 'pred' arguments instead of 'model'.
obs	alternatively to 'model' and together with 'pred', a numeric vector of observed presences (1) and absences (0) of a binary response variable. Alternatively (and if 'pred' is a 'SpatRaster'), a two-column matrix or data frame containing, respectively, the x (longitude) and y (latitude) coordinates of the presence points, in which case the 'obs' vector will be extracted with ptsrast2obspred. This argument is ignored if 'model' is provided.
pred	alternatively to 'model' and together with 'obs', a vector with the corresponding predicted values of presence probability, habitat suitability, environmental favourability or alike. Must be of the same length and in the same order as 'obs'. Alternatively (and if 'obs' is a set of point coordinates), a 'SpatRaster' map of the predicted values for the entire evaluation region, in which case the 'pred' vector will be extracted with ptsrast2obspred. This argument is ignored if 'model' is provided.
threshMethod	Criterion under which to compute the threshold. Run modEvAmethods("getThreshold") for available options.
interval	Argument to pass to optiThresh indicating the interval between the thresholds to test. The default is 0.01. Smaller values may provide more precise results but take longer to compute.
quant	Numeric value indicating the proportion of presences to discard if threshMethod="MTP" (minimum training presence). With the default value 0, MTP will be the threshold at which all observed presences are classified as such; with e.g. quant=0.05, MTP will be the threshold at which 5% presences will be classified as absences.
na.rm	Logical value (default TRUE) indicating whether NA values should be ignored.
verbosity	integer indicating the amount of messages to display. Defaults to the maximum implemented; lower numbers (down to 0) decrease the number of messages.
pbg	logical value to pass to inputMunch indicating whether to use presence/background (rather than presence/absence) data. Default FALSE.

Details

Several criteria have been proposed for selecting a threshold with which to convert continuous model predictions (of presence probability, habitat suitability or alike) into binary predictions of presence or absence. Such threshold is required for computing threshold-based model evaluation metrics, such as those in threshMeasures. This function implements a few of these threshold selection criteria, including those outlined in Liu et al. (2005, 2013) and a couple more:

- "preval", "trainPrev": prevalence (proportion of presences) in the supplied 'model' or 'obs'

- "meanPred": mean predicted value in the supplied 'model' or 'pred'

- "midPoint": median predicted value in the supplied 'model' or 'pred'

- "maxKappa": threshold that maximizes Cohen's kappa

- "maxCCR", "maxOA", "maxOPS": threshold that maximizes the Correct Classification Rate, aka Overall Accuracy, aka Overall Prediction Success

- "maxF": threshold that maximizes the F value

- "maxSSS": threshold that maximizes the sum of sensitivity and specificity

- "maxTSS": threshold that maximizes the True Skill Statistic

- "maxSPR": threshold that maximizes the sum of precision and recall

- "minDSS": threshold that minimizes the difference between sensitivity and specificity

- "minDPR": threshold that minimizes the difference between precision and recall

- "minD01": threshold that minimizes the distance between the ROC curve and the 0,1 point

- "minD11": threshold that minimizes the distance between the PR curve and the 1,1 point

- "equalPrev": predicted and observed prevalence equalization

- "MTP": minimum training presence, or the lowest predicted value where presence is recorded in 'obs' or 'model'. Optionally, with the 'quant' argument, this threshold leaves out predicted values lower than the value for the lowest specified proportion of presences

Value

This function returns a numeric value indicating the threshold selected under the specified 'threshMethod'.

Author(s)

A. Marcia Barbosa

References

Liu C., Berry P.M., Dawson T.P. & Pearson R.G. (2005) Selecting thresholds of occurrence in the prediction of species distributions. Ecography 28: 385-393

Liu C., White M. & Newell G. (2013) Selecting thresholds for the prediction of species occurrence with presence-only data. Journal of Biogeography 40: 778-789

See Also

optiThresh, optiPair, threshMeasures

Examples

```
# load sample models:
data(rotif.mods)

# choose a particular model to play with:
mod <- rotif.mods$models[[1]]

getThreshold(model = mod, threshMethod = "maxTSS")


# you can also use getThreshold with vectors of observed and predicted values
```

```
# instead of with a model object:

presabs <- mod$y
prediction <- mod$fitted.values

getThreshold(obs = presabs, pred = prediction, threshMethod = "maxTSS")

getThreshold(obs = presabs, pred = prediction, threshMethod = "MTP")

getThreshold(obs = presabs, pred = prediction, threshMethod = "MTP",
quant = 0.05)


# 'obs' can also be a table of presence point coordinates
# and 'pred' a SpatRaster of predicted values
```

`HLfit` *Hosmer-Lemeshow goodness of fit*

Description

This function calculates a model's calibration performance (reliability) with the Hosmer & Lemeshow goodness-of-fit statistic, which compares predicted probability to observed occurrence frequency at each portion of the probability range.

Usage

```
HLfit(model = NULL, obs = NULL, pred = NULL, bin.method,
n.bins = 10, fixed.bin.size = FALSE, min.bin.size = 15,
min.prob.interval = 0.1, quantile.type = 7, simplif = FALSE,
verbosity = 2, alpha = 0.05, plot = TRUE, plot.values = TRUE,
values.col = c("grey40", "grey40", "black"),
plot.bin.size = TRUE, xlab = "Predicted probability",
ylab = "Observed prevalence", na.rm = TRUE, rm.dup = FALSE, ...)
```

Arguments

`model`	optional binary-response model object of class "glm", "gam", "gbm", "randomForest" or "bart". If this argument is provided, 'obs' and 'pred' will be extracted with `mod2obspred`. Alternatively, you can input the 'obs' and 'pred' arguments instead of 'model'.
`obs`	alternatively to 'model' and together with 'pred', a numeric vector of observed presences (1) and absences (0) of a binary response variable. Alternatively (and if 'pred' is a 'SpatRaster'), a two-column matrix or data frame containing, respectively, the x (longitude) and y (latitude) coordinates of the presence points, in which case the 'obs' vector will be extracted with `ptsrast2obspred`. This argument is ignored if 'model' is provided.

pred	alternatively to 'model' and together with 'obs', a vector with the corresponding predicted values of presence probability, habitat suitability, environmental favourability or alike. Must be of the same length and in the same order as 'obs'. Alternatively (and if 'obs' is a set of point coordinates), a 'SpatRaster' map of the predicted values for the entire evaluation region, in which case the 'pred' vector will be extracted with ptsrast2obspred. This argument is ignored if 'model' is provided.
bin.method	argument to pass to getBins specifying the method for grouping the records into bins within which to compare predicted probability to observed prevalence; type modEvAmethods("getBins") for available options, and see Details for more information.
n.bins	argument to pass to getBins (default 10) specifying the number of bins to use if bin.method = n.bins or bin.method = quantiles.
fixed.bin.size	argument to pass to getBins, a logical value (default FALSE) indicating whether to force bins to have (approximately) the same size.
min.bin.size	argument to pass to getBins specifying the minimum number of records in each bin. The default is 15, the minimum required for accurate comparisons of within-bin proportions (Jovani & Tella 2006, Jimenez-Valverde et al. 2013).
min.prob.interval	argument to pass to getBins specifying the minimum interval (range) of probability values within each bin. The default is 0.1.
quantile.type	argument to pass to quantile specifying the algorithm to use if bin.method = "quantiles". The default is 7 (the quantile default in R), but check out other types, e.g. 3 (used by SAS), 6 (used by Minitab and SPSS) or 5 (appropriate for deciles, which correspond to the default n.bins = 10).
simplif	logical (default FALSE), wheter to perform a faster simplified version returning only the basic statistics.
verbosity	integer specifying the amount of messages or warnings to display. Defaults to the maximum implemented; lower numbers (down to 0) decrease the number of messages.
alpha	alpha value for confidence intervals if plot = TRUE.
plot	logical (default TRUE), whether to produce a plot of the results.
plot.values	logical (default TRUE), whether to report measure values in the plot.
values.col	character vector of length 1 or 3 specifying the colour(s) for the values in the plot.
plot.bin.size	logical (default TRUE), whether to report bin sizes in the plot. Sizes below 15 are coloured red, flagging less meaningful predicted-observed comparisons within bins (Jovani & Tella 2006, Jimenez-Valverde et al. 2013).
xlab	label for the x axis.
ylab	label for the y axis.
na.rm	Logical (default TRUE) indicating whether missing values should be ignored in computations.
rm.dup	If TRUE and if 'pred' is a SpatRaster and if there are repeated points within the same pixel, a maximum of one point per pixel is used to compute the presences. See examples in ptsrast2obspred. The default is FALSE.

...	further arguments to pass to the `plot` function.

Details

Most of the commonly used measures for evaluating model performance focus on the discrimination or the classification capacity, i.e., how well the model is capable of distinguishing or classifying presences and absences (often after the model's continuous predictions of presence probability or alike are converted to binary predictions of presence or absence). However, there is another important facet of model evaluation: calibration or reliability, i.e., the relationship between predicted probability and observed occurrence frequency (Pearce & Ferrier 2000; Jimenez-Valverde et al. 2013). The `HLfit` function measures model reliability with the Hosmer & Lemeshow goodness-of-fit statistic (Hosmer & Lemeshow 1980).

Note that this statistic has strong limitations and caveats (see e.g. http://www.statisticalhorizons.com/hosmer-lemeshow, Allison 2014), mainly due to the need to group the values into bins within which to compare probability and prevalence, and the strong influence of the binning method on the results. The 'HLfit' function can use several binning methods, which are implemented and roughly explained in the `getBins` function and can be accessed by typing 'modEvAmethods("getBins")'. You should try 'HLfit' with different binning methods to see if the results are robust.

Value

`HLfit` returns a list with the following components:

`bins.table`	a data frame of the obtained bins and the values resulting from the Hosmer-Lemeshow goodness-of-fit analysis.
`chi.sq`	the value of the Chi-squared test, i.e. the Hosmer-Lemeshow (HL) value, which is larger for larger differences between predicted and observed presence proportions.
`DF`	the number of degrees of freedom.
`p.value`	the p-value of the Hosmer-Lemeshow test. Note that this is one of those tests for which higher p-values are better (i.e., less significant predicted-observed differences).
`RMSE`	the root mean squared error.
`HLI`	[EXPERIMENTAL, under review] the inverse standardized HL (Chi-squared) value, bounded in [0, 1] and larger for smaller predicted-observed differences (i.e. for better-calibrated models).

Note

The 4 lines of code from "observed" to "p.value" were adapted from the 'hosmerlem' function available at http://www.stat.sc.edu/~hitchcock/diseaseoutbreakRexample704.txt. The plotting code was loosely based on the `calibration.plot` function in package **PresenceAbsence**. `HLfit` still needs some code simplification, and may fail for some datasets and binning methods. Fixes are being applied. Feedback is welcome.

References

Allison P.D. (2014) Measures of Fit for Logistic Regression. SAS Global Forum, Paper 1485

Hosmer D.W. & Lemeshow S. (1980) A goodness-of-fit test for the multiple logistic regression model. Communications in Statistics, A10: 1043-1069

Jimenez-Valverde A., Acevedo P., Barbosa A.M., Lobo J.M. & Real R. (2013) Discrimination capacity in species distribution models depends on the representativeness of the environmental domain. Global Ecology and Biogeography 22: 508-516

Jovani R. & Tella J.L. (2006) Parasite prevalence and sample size: misconceptions and solutions. Trends in Parasitology 22: 214-218

Pearce J. & Ferrier S. (2000) Evaluating the Predictive Performance of Habitat Models Developed using Logistic Regression. Ecological Modeling, 133: 225-245

See Also

getBins, Miller, Boyce

Examples

```
# load sample models:

data(rotif.mods)


# choose a particular model to play with:

mod <- rotif.mods$models[[1]]


# try HLfit using different binning methods:

HLfit(model = mod, bin.method = "round.prob",
main = "HL GOF with round.prob (n=10)")

HLfit(model = mod, bin.method = "prob.bins",
main = "HL GOF with prob.bins (n=10)")

HLfit(model = mod, bin.method = "size.bins",
main = "HL GOF with size.bins (min size=15)")

HLfit(model = mod, bin.method = "size.bins", min.bin.size = 30,
main = "HL GOF with size.bins min size 30")

HLfit(model = mod, bin.method = "n.bins",
main = "HL GOF with 10 bins")

HLfit(model = mod, bin.method = "n.bins", fixed.bin.size = TRUE,
main = "HL GOF with 10 bins of fixed size")

HLfit(model = mod, bin.method = "n.bins", n.bins = 20,
main = "HL GOF with 20 bins")
```

```
HLfit(model = mod, bin.method = "quantiles",
main = "HL GOF with quantile bins (n=10)")

HLfit(model = mod, bin.method = "quantiles", n.bins = 20,
main = "HL GOF with quantile bins (n=20)")


# you can also use 'predPlot' with vectors of observed and predicted values
# instead of a model object:

presabs <- mod$y
prediction <- mod$fitted.values

HLfit(obs = presabs, pred = prediction, bin.method = "round.prob")


# 'obs' can also be a table of presence point coordinates
# and 'pred' a SpatRaster of predicted values
```

`inputMunch` *Munch inputs into 'obs' and 'pred' vectors*

Description

This function is used internally by many other functions in this package to check and extract the 'obs' and 'pred' vectors from the user inputs.

Usage

```
inputMunch(model = NULL, obs = NULL, pred = NULL, rm.dup = FALSE, na.rm = FALSE,
pbg = FALSE, verbosity = 2)
```

Arguments

`model`	optional binary-response model object of class "glm", "gam", "gbm", "randomForest" or "bart". If this argument is provided, 'obs' and 'pred' will be extracted with `mod2obspred`. Alternatively, you can input the 'obs' and 'pred' arguments (e.g. for external test data) instead of 'model'.
`obs`	alternatively to 'model' and together with 'pred', a numeric vector of observed presences (1) and absences (0) of a binary response variable. Alternatively (and if 'pred' is a 'SpatRaster'), a two-column matrix or data frame containing, respectively, the x (longitude) and y (latitude) coordinates of the presence points, in which case the 'obs' vector will be extracted with `ptsrast2obspred`. This argument is ignored if 'model' is provided.
`pred`	alternatively to 'model' and together with 'obs', a vector with the corresponding predicted values of presence probability, habitat suitability, environmental favourability or alike. Must be of the same length and in the same order as 'obs'.

Alternatively (and if 'obs' is a set of point coordinates), a 'SpatRaster' map of the predicted values for the entire evaluation region, in which case the 'pred' vector will be extracted with ptsrast2obspred. This argument is ignored if 'model' is provided.

`rm.dup` logical argument to be passed to ptsrast2obspred indicating whether repeated points within the same pixel should be removed. The default is FALSE.

`na.rm` logical argument indicating whether to remove (with a warning saying how many) rows with NA in any of the resulting 'obs' or 'pred' values. The default is FALSE.

`pbg` logical value indicating whether to output presence/background (rather than presence/absence) data. If set to TRUE, the 'pred' values at presences ('obs' = 1) are added to the background ('obs' = 0). Similar to what is done in presence/background modelling algorithms such as Maxent (e.g. analogous to the 'addsamplestobackground' argument of maxnet::maxnet). Experimental! Default FALSE.

`verbosity` integer value indicating the amount of messages to display. Defaults to 2, for the maximum amount of messages.

Value

This function returns a data frame containing the 'obs' and 'pred' values, or an error message if inputs are not as required.

Author(s)

A. Marcia Barbosa

`logLike` *Log-likelihood*

Description

This function computes the log-likelihood of a model, given a model object or a set of observed and predicted values (or presence points and raster predictions).

Usage

```
logLike(model = NULL, obs = NULL, pred = NULL, na.rm = TRUE, plot = TRUE, verbosity = 2)
```

Arguments

`model` optional model object of class implemented in mod2obspred. If this argument is provided, 'obs' and 'pred' will be extracted with that function. Alternatively, you can input the 'obs' and 'pred' arguments instead of 'model'.

obs	alternatively to 'model' and together with 'pred', a numeric vector of observed values of the response variable. Alternatively (and if 'pred' is a 'SpatRaster'), a two-column matrix or data frame containing, respectively, the x (longitude) and y (latitude) coordinates of presence points, in which case the 'obs' vector of presences and absences will be extracted with ptsrast2obspred. This argument is ignored if 'model' is provided.
pred	alternatively to 'model' and together with 'obs', a vector with the corresponding predicted values, of the same length and in the same order as 'obs'. Alternatively (and if 'obs' is a set of point coordinates), a 'SpatRaster' map of the predicted values for the entire evaluation region, in which case the 'pred' vector will be extracted with ptsrast2obspred. This argument is ignored if 'model' is provided.
na.rm	logical value indicating whether missing values should be ignored in computations. Defaults to TRUE.
plot	logical value indicating whether to plot the sorted values in log() (see Details). The default is TRUE.
verbosity	integer specifying the amount of messages to display. Defaults to the maximum implemented; lower numbers (down to 0) decrease the number of messages.

Details

The log-likelihood can be a measure of model calibration (Lawson et al. 2014, Fithian et al. 2015, Fletcher & Fortin 2018). The higher the value, the better the predictions fit the observations.

The log-likelihood is computed as sum(log(pred*obs+(1-pred)*(1-obs))) (Fletcher & Fortin 2018, p. 234). If plot=TRUE, a plot is shown with the values within sum(), both within and outside log().

In the first instance of 'pred' in the formula above, a very small constant is added to the 'pred' values of zero, because the logarithm of zero is undefined. This added constant is smaller (i.e. closer to the original value of zero) here than in the R code accompanying Fletcher & Fortin (2018), specifically 2e-16 rather than 0.001, which may justify some differences in the results when there are predictions of exactly zero. In any case, whatever constant is used, it should be the same across the models being compared.

Value

This function returns a numeric value indicating the log-likelihood of the input model predictions given the input observations.

Author(s)

A. Marcia Barbosa

References

Fithian, W., Elith, J., Hastie, T., Keith, D.A., 2015. Bias correction in species distribution models: pooling survey and collection data for multiple species. Methods in Ecology and Evolution 6:424-438. https://doi.org/10.1111/2041-210X.12242

Fletcher R. & Fortin M.-J. (2018) Spatial Ecology and Conservation Modeling. Applications with R. Springer Nature Switzerland. Cham: 532 pp. https://www.fletcherlab.com/spatial-ecology-conservation-modeli

Lawson, C.R., Hodgson, J.A., Wilson, R.J., Richards, S.A., 2013. Prevalence, thresholds and the performance of presence-absence models. Methods in Ecology and Evolution 5, 54-64. https://doi.org/10.1111/2041-210X.12123

See Also

logLik, Miller, HLfit, Boyce, RMSE, Dsquared, pseudoRsq

Examples

```
# load sample models:
data(rotif.mods)

# choose a particular model to play with:
mod <- rotif.mods$models[[1]]

logLike(model = mod)


# you can also use logLike with vectors of observed and predicted values
# instead of with a model object:

obs <- mod$y
pred <- mod$fitted.values

logLike(obs = obs, pred = pred)


# 'obs' can also be a table of presence point coordinates
# and 'pred' a SpatRaster of predicted values
```

---

lollipop *Lollipop chart*

Description

This function creates a lollipop chart (by default), a points chart (if `sticks = FALSE`) or a point-range chart (if `sticks` are confidence limits) from a (named) numeric vector.

Usage

```
lollipop(x, names = NULL, ymin = 0, ylim = "auto0", sticks = TRUE,
col = "steelblue4", grid = TRUE, cex = 1, cex.axis = 1, las = 2,
horiz = FALSE, bold = FALSE, ...)
```

Arguments

x	a numeric vector.
names	a vector of the same length as 'x' with the names to be plotted below the lollipops. If this argument is left NULL and 'x' has names, then these will be used.
ymin	numeric value for the lower limit of the y axis (default 0). If set to NA, the minimum of 'x' will be used. This argument is deprecated, as it is now superseded by, and should be replaced with, 'ylim' (below).
ylim	either a numeric vector of length 2 specifying the limits (minimum, maximum) for the y axis; or "auto" to fit the y axis to the minimum and maximum existing values in 'x'; or "auto0" (the new default) to fit the top of the y axis to the maximum existing value, and the bottom to zero or the minimum existing value (whichever is lower). If horiz is TRUE, these limits refer instead to the x axis.
sticks	either a logical value indicating whether the sticks of the lollipops (lines from zero to each value) should be drawn (default TRUE); or a two-column matrix or dataframe with one row for each element in x, with the minimum and maximum values for the error bars or confidence intervals (in which case a point-range chart is drawn).
col	colour for the lollipops.
grid	logical value (default TRUE) indicating whether or not to add a grid to the plot.
cex	numeric value indicating the size of the lollipops. Will be passed as 'cex' to points and as 'lwd' to arrows (the lines or lollipop sticks).
cex.axis	numeric value to pass to axis and to mtext indicating the size of the x and y axis labels.
las	argument to pass to par indicating the orientation of the axis labels.
horiz	logical value (default FALSE) indicating whether the plot should be drawn horizontally rather than vertically.
bold	logical vector (recycled if shorter than x, default FALSE) indicating which name labels should appear in bold font. Can be used e.g. to highlight significant values (see e.g. plotCoeffs).
...	additional arguments that can be passed to plot, e.g. 'main', 'xlab' or 'ylab'.

Details

According to modern data viz recommendations, lollipop charts (or instead point charts) are generally a better alternative to bar charts, as they reduce the visual distortion caused by the length of the bars, making it easier to visually compare the values.

Value

NULL, with a plot as side effect.

Author(s)

A. Marcia Barbosa

See Also

plotCoeffs; barplot; ggplot2::geom_pointrange()

Examples

```
par(mar = c(6, 3, 2, 1))

lollipop(mtcars[,1], names = rownames(mtcars), las = 2,
ylab = names(mtcars)[1], cex.axis = 0.6)

lollipop(mtcars[,1], names = rownames(mtcars), las = 2,
ylab = names(mtcars)[1], cex.axis = 0.6, ylim = NA)


par(mar = c(3, 6, 2, 1))

lollipop(mtcars[,1], names = rownames(mtcars), las = 2,
xlab = "Miles/gallon", cex.axis = 0.6, main = "Lollipop chart of mtcars",
horiz = TRUE)
```

MESS *Multivariate Environmental Similarity Surfaces based on a data frame*

Description

This function performs the MESS analysis of Elith et al. (2010) to determine the extent of the environmental differences between model training and model projection (extrapolation) data. It is applicable to variables in a matrix or data frame.

Usage

```
MESS(V, P, id.col = NULL, verbosity = 2)
```

Arguments

V	a matrix or data frame containing the variables (one in each column) in the training dataset.
P	a matrix or data frame containing the same variables in the area to which the model(s) will be projected. Variables (columns) must be in the same order as in V, and colnames(P) must exist.
id.col	optionally, the index number of a column containing the row identifiers in P. If provided, this column will be excluded from MESS calculations but included in the output.
verbosity	Integer number indicating the amount of messages to display while computing the results. The default is to display all messages. Set verbosity=0 for no messages.

Details

When model predictions are projected into regions, times or spatial resolutions not analysed in the training data, it may be important to measure the similarity between the new environments and those in the training sample (Elith et al. 2010), as models are not so reliable when predicting outside their domain (Barbosa et al. 2009). The Multivariate Environmental Similarity Surfaces (MESS) analysis measures the similarity in the analysed variables between any given locality in the projection dataset and the localities in the reference (training) dataset (Elith et al. 2010).

MESS analysis is implemented in the MAXENT software (Phillips et al. 2006) and in the **dismo** R package, but there it requires input variables in raster format. This implies not only the use of complex spatial data structures, but also that the units of analysis are rectangular pixels, whereas we often need to model distribution data recorded on less regular units (e.g. provinces, river basins), or on equal-area cells that are not necessarily rectangular (e.g. UTM cells, equal-area hexagons or other geometric shapes). The MESS function computes this analysis for variables in a data frame, where localities (in rows) may be of any size or shape.

Value

The function returns a data frame with the same column names as `P`, plus a column named `TOTAL`, quantifying the similarity between each point in the projection dataset and those in the reference dataset. Negative values indicate localities that are environmentally dissimilar from the reference region. The last column, `MoD`, indicates which of the column names of P corresponds to the most dissimilar variable, i.e., the limiting factor or the variable that drives the MESS in that locality (Elith et al. 2010).

Note

Newer and apparently more complete methods for analysing environmental dissimilarities have been developed, such as extrapolation detection (ExDet; Mesgaran et al. 2014) and Mobility-Oriented Parity analysis (MOP; Owens et al. 2013).

Author(s)

Alberto Jimenez-Valverde, A. Marcia Barbosa

References

Barbosa A.M., Real R. & Vargas J.M. (2009) Transferability of environmental favourability models in geographic space: the case of the Iberian desman (Galemys pyrenaicus) in Portugal and Spain. Ecological Modelling 220: 747-754

Elith J., Kearney M. & Phillips S. (2010) The art of modelling range-shifting species. Methods in Ecology and Evolution 1: 330-342

Mesgaran M.B., Cousens R.D. & Webber B.L. (2014) Here be dragons: a tool for quantifying novelty due to covariate range and correlation change when projecting species distribution models. Diversity and Distributions, 20: 1147-1159

Owens H.L., Campbell L.P., Dornak L.L., Saupe E.E., Barve N., Soberon J., Ingenloff K., Lira-Noriega A., Hensz C.M., Myers C.E. & Peterson A.T. (2013) Constraints on interpretation of ecological niche models by limited environmental ranges on calibration areas. Ecological Modelling, 263: 10-18

Phillips S.J., Anderson R.P. & Schapire R.E. (2006) Maximum entropy modeling of species geographic distributions. Ecological Modelling 190: 231-259

See Also

OA; mess in packages **dismo** and **predicts**; ecospat.climan in package **ecospat**; kuenm_mop and kuenm_mmop in package **kuenm**

Examples

```
## Not run:
# load package 'fuzzySim' and its sample data:
require(fuzzySim)
data(rotif.env)


# add a column specifying the hemisphere:

unique(rotif.env$CONTINENT)

rotif.env$HEMISPHERE <- "Eastern"

rotif.env$HEMISPHERE[rotif.env$CONTINENT %in%
c("NORTHERN_AMERICA", "SOUTHERN_AMERICA")] <- "Western"

head(rotif.env)


# perform a MESS analysis
# suppose you'll extrapolate models from the Western hemisphere (Americas)
# to the Eastern hemisphere (rest of the world):

names(rotif.env)  # variables are in columns 5:17

west <- subset(rotif.env, HEMISPHERE == "Western", select = 5:17)
east <- subset(rotif.env, HEMISPHERE == "Eastern", select = 5:17)
east.with.ID <- subset(rotif.env, HEMISPHERE == "Eastern",
select = c(1, 5:17))

head(east)
head(east.with.ID)  # ID is in column 1

mess <- MESS(V = west, P = east)
mess.with.ID <- MESS(V = west, P = east.with.ID, id.col = 1)

head(mess)
head(mess.with.ID)

range(mess[ , "TOTAL"])

## End(Not run)
```

`Miller` *Miller's model calibration satistics*

Description

This function calculates Miller's (1991) calibration statistics for presence probability model predictions – namely, the intercept and slope of a logistic regression of the response variable on the logit of predicted probabilities. Optionally and by default, it also plots the corresponding regression line over the reference diagonal (identity line). If the model is well calibrated, the line should lie along (or at least be nearly parallel to) the reference diagonal, i.e. the slope should ideally equal 1.

Usage

```
Miller(model = NULL, obs = NULL, pred = NULL, plot = TRUE,
line.col = "steelblue4", diag = TRUE, diag.col = "lightblue3",
plot.values = TRUE, digits = 2, values.col = c("grey40", "#8B4C4C", "black"),
xlab = "", ylab = "", main = "Miller calibration", na.rm = TRUE, rm.dup = FALSE,
verbosity = 2, ...)
```

Arguments

`model`	optional binary-response model object of class "glm", "gam", "gbm", "randomForest" or "bart". If this argument is provided, 'obs' and 'pred' will be extracted with `mod2obspred`. Alternatively, you can input the 'obs' and 'pred' arguments instead of 'model'.
`obs`	alternatively to 'model' and together with 'pred', a numeric vector of observed presences (1) and absences (0) of a binary response variable. Alternatively (and if 'pred' is a 'SpatRaster'), a two-column matrix or data frame containing, respectively, the x (longitude) and y (latitude) coordinates of the presence points, in which case the 'obs' vector will be extracted with `ptsrast2obspred`. This argument is ignored if 'model' is provided.
`pred`	alternatively to 'model' and together with 'obs', a vector with the corresponding predicted values of presence probability, habitat suitability, environmental favourability or alike. Must be of the same length and in the same order as 'obs'. Alternatively (and if 'obs' is a set of point coordinates), a 'SpatRaster' map of the predicted values for the entire evaluation region, in which case the 'pred' vector will be extracted with `ptsrast2obspred`. This argument is ignored if 'model' is provided.
`plot`	logical (default TRUE), whether or not to produce a plot of the Miller regression line.
`line.col`	colour for the Miller regression line in the plot.
`diag`	logical (default TRUE), whether or not to add the reference diagonal to the plot.
`diag.col`	colour for the reference diagonal in the plot.

plot.values	logical (default TRUE), whether or not to show in the plot the values of the Miller slope, the absolute difference between the slope and the optimal value of 1, and the Unit-Slope Proximity [the latter is EXPERIMENTAL - see Value section].
digits	integer value (default 2) indicating the number of digits to which the values in the plot should be rounded
values.col	character vector of length 1 or 3 specifying the colour(s) for the values in the plot.
xlab	label for the x axis.
ylab	label for the y axis.
main	title for the plot.
na.rm	Logical value (default TRUE) indicating whether missing values should be ignored in computations.
rm.dup	If TRUE and if 'pred' is a SpatRaster and if there are repeated points within the same pixel, a maximum of one point per pixel is used to compute the presences. See examples in ptsrast2obspred. The default is FALSE.
verbosity	integer specifying the amount of messages to display. Defaults to the maximum implemented; lower numbers (down to 0) decrease the number of messages.
...	additional arguments to pass to plot.

Details

Calibration or reliability assesses how a model's predicted probabilities relate to observed species prevalence or proportion of presences in the modelled data (Pearce & Ferrier 2000; Wintle et al. 2005; Franklin 2010). If predictions are perfectly calibrated, the slope will equal 1 and the intercept will equal 0, so the model's calibation line will perfectly overlap with the reference diagonal or identity line.

Note that Miller's statistics assess the model globally: a model is well calibrated if the average of all predicted probabilities equals the proportion of presences in the modelled data. For logistic regression models, perfect calibration is always attained on the same training data used for building the model (Miller 1991); in this case, Miller's calibration statistics are only useful when projecting a model outside those training data.

Calibration can be separated into two measurable components, bias and spread, and a third component, unexplained error. Bias describes a consistent overestimate or underestimate of presence probability, which is reflected by a Miller intercept above or below 0 (i.e., a model line above or below the reference diagonal). Spread describes a departure of the model line from the 45-degree slope. A slope greater than 1 indicates that predicted values above 0.5 are underestimating, and predicted values below 0.5 are overestimating, the probability of presence. A slope smaller than 1 (while greater than 0) implies that predicted values below 0.5 are underestimating, and values above 0.5 are overestimating, the probability of presence (Pearce & Ferrier 2000). A Miller slope very different from 1 indicates a poorly calibrated model. Baquero et al. (2021) proposed that a value between 0.5 and 1.5 (i.e., absolute slope difference up to 0.5) can be considered not very different from 1. The unexplained error component can be assessed, though only in part, through residual analysis (Miller, 1991; Pearce & Ferrier, 2000).

While Miller's calibration statistics were originally conceived for generalized linear models with binomial distribution and logit link (Miller, 1991), they may also apply to other models that estimate

presence probability, including those that use different link functions such as probit or cloglog. Regardless of how they get there, these models attempt to estimate presence probabilities as well calibrated as possible, and the logit is the canonical link for the Bernoulli distribution which is appropriate for a binary response variable (Jane Elith, pers. comm.). Indeed, the Miller slope is visibly worse if those other link functions are used for computing it.

Miller's calibration slope (though not the intercept) is also adequate to assess the calibration of other predictions related to presence probability, such as suitability and favourability (see e.g. 'Fav' function in the **fuzzySim** package). Indeed, the slope is the same for presence probability and its corresponding favourability value (see Examples, bottom).

Value

This function returns a named list of four numeric values:

`intercept`	the Miller calibration intercept.
`slope`	the Miller calibration slope.
`slopeDiff`	the absolute difference between the obtained slope and the optimal value of 1.
`MCA`	[EXPERIMENTAL, under review] the Miller Calibration Alignment: a standardized, monotonic non-linear measure, bounded between 0 and 1, of how aligned (parallel) the calibration line is to the reference diagonal, i.e. how close the slope is to the optimal value of 1.

If `plot = TRUE`, a plot will be produced with the model calibration line, optionally (if diag = TRUE) over the reference diagonal, and also optionally (if plot.values = TRUE) with output values printed on it.

References

Baquero R.A., Barbosa A.M., Ayllon D., Guerra C., Sanchez E., Araujo M.B. & Nicola G.G. (2021) Potential distributions of invasive vertebrates in the Iberian Peninsula under projected changes in climate extreme events. Diversity and Distributions, 27(11): 2262-2276

Franklin, J. (2010) Mapping Species Distributions: Spatial Inference and Prediction. Cambridge University Press, Cambridge

Miller M.E., Hui S.L. & Tierney W.M. (1991) Validation techniques for logistic regression models. Statistics in Medicine, 10: 1213-1226

Pearce J. & Ferrier S. (2000) Evaluating the predictive performance of habitat models developed using logistic regression. Ecological Modelling, 133: 225-245

Wintle B.A., Elith J. & Potts J.M. (2005) Fauna habitat modelling and mapping: A review and case study in the Lower Hunter Central Coast region of NSW. Austral Ecology, 30: 719-738

See Also

`HLfit`, `Dsquared`, `pseudoRsq`, `Boyce`

Examples

```
# load sample models:
data(rotif.mods)

# choose a particular model to play with:
mod <- rotif.mods$models[[1]]

Miller(model = mod)
Miller(model = mod, plot.values = FALSE, main = "Model calibration line")


# you can also use Miller with vectors of observed and predicted values
# instead of a model object:

Miller(obs = mod$y, pred = mod$fitted.values)


# 'obs' can also be a table of presence point coordinates
# and 'pred' a SpatRaster of predicted values


# Miller slope can also apply to predictions other than probability:

## Not run:
# (the following code requires the 'fuzzySim' pkg)

Miller(obs = mod$y, pred = mod$fitted.values)  # probability

fav <- fuzzySim::Fav(model = mod)  # favourability

Miller(obs = mod$y, pred = fav)  # same slope, different intercept

## End(Not run)
```

MillerCalib *Miller's model calibration satistics*

Description

This function is DEPRECATED, please use Miller instead.

Usage

```
MillerCalib(model = NULL, obs = NULL, pred = NULL, plot = TRUE,
line.col = "darkblue", diag = TRUE, diag.col = "lightblue3",
plot.values = TRUE, digits = 2, xlab = "", ylab = "",
main = "Miller calibration", na.rm = TRUE, rm.dup = FALSE, verbosity = 2, ...)
```

Arguments

model	see Miller
obs	see Miller
pred	see Miller
plot	see Miller
line.col	see Miller
diag	see Miller
diag.col	see Miller
plot.values	see Miller
digits	see Miller
xlab	see Miller
ylab	see Miller
main	see Miller
na.rm	see Miller
rm.dup	see Miller
verbosity	see Miller
...	see Miller

Details

see Miller

Value

see Miller

Author(s)

A. Marcia Barbosa

References

see Miller

See Also

Miller

Examples

```
# This function is DEPRECATED, please see ?Miller instead.
```

mod2obspred *Extract observed and predicted, or predictor, values from a model object.*

Description

This function takes a model object and returns the observed and (optionally) the predicted (fitted) values from that model; or the values of the predictors in the model.

Usage

```
mod2obspred(model, obs.only = FALSE, x.only = FALSE)
```

Arguments

model	a model object of class "glm", "gam", "gbm", "GBMFit", "randomForest" or "bart" from which the response variable and fitted (predicted) values can be extracted. Note that, for "randomForest" models, only the out-of-bag prediction is available from the model object (see ?randomForest::predict.randomForest), so here you'll get different results if you provide 'model' or the modelled 'obs' (and corresponding 'pred') values.
obs.only	logical value (default FALSE) indicating whether only 'obs' should be returned. Reduces computation when 'pred' is not needed. Used e.g. by prevalence).
x.only	logical value (default FALSE) indicating whether only 'x' or 'data' (the predictor values) should be returned. Used e.g. by varImp if imp.type = "permutation", or by clampVars() in package **fuzzySim** if 'ref' is a model object.

Value

A data frame with one column containing the observed and (if obs.only=FALSE, the default) another column containing the predicted values from 'model'; or a data frame with the 'x', 'data' or predictor values from 'model' (if x.only=TRUE).

Author(s)

A. Marcia Barbosa

See Also

prevalence

Examples

```
data(rotif.mods)
mod <- rotif.mods$models[[1]]

obspred <- mod2obspred(mod)
head(obspred)
```

```
obs <- mod2obspred(mod, obs.only = TRUE)
head(obs)

data <- mod2obspred(mod, x.only = TRUE)
head(data)
```

`modEvAmethods` *Methods implemented in modEvA functions*

Description

This function allows retrieving the methods available for some of the functions in modEvA, such as `similarity`, `threshMeasures`, `optiThresh`, `multModEv`, `getThreshold` and `getBins`.

Usage

```
modEvAmethods(fun)
```

Arguments

`fun`	a character vector of length 1 specifying the name (in quotes) of the function for which to obtain the available methods. Must be one of "threshMeasures", "optiThresh", "multModEv", "getThreshold" or "getBins".

Value

a character vector of the available methods for the specified function.

Author(s)

A. Marcia Barbosa

See Also

`threshMeasures`, `optiThresh`, `getBins`, `multModEv`

Examples

```
modEvAmethods("threshMeasures")

modEvAmethods("multModEv")

modEvAmethods("optiThresh")

modEvAmethods("getBins")

modEvAmethods("similarity")
```

multModEv *Multiple model evaluation*

Description

If you have a list of GLM model objects (created, e.g., with the multGLM function of the 'fuzzySim' R-Forge package), or a data frame with presence-absence data and the corresponding predicted values for a set of species, you can use the multModEv function to get a set of evaluation measures for all models simultaneously, as long as they all have the same sample size.

Usage

```
multModEv(models = NULL, obs.data = NULL, pred.data = NULL,
measures = modEvAmethods("multModEv"), standardize = FALSE,
thresh = NULL, bin.method = NULL, verbosity = 0, ...)
```

Arguments

models	optional list of model object(s) of class "glm", all applied to the same data set. Evaluation is based on the cases included in the models.
obs.data	alternatively to 'models' and together with 'pred.data', a data frame with observed (training or test) binary data (1 for presence, 0 for absence of records). This argument is ignored if 'models' is provided.
pred.data	alternatively to 'models' and together with 'obs.data', a data frame with the corresponding predicted (training or test) values, with both rows and columns in the same order as in 'obs.data'. This argument is ignored if 'models' is provided. Note that some calibration measures (e.g. HLfit, Miller) assume that predictions represent probability, while others (e.g. Boyce) assume suitability or favourability.
measures	character vector of the evaluation measures to calculate. The default is all implemented measures, which you can check by typing 'modEvAmethods("multModEv")'. But beware: calibration measures (i.e., HL and Miller) are only valid if your predicted values reflect actual presence probability (not favourability, habitat suitability or others); you should exclude them otherwise.
standardize	logical, whether to standardize measures that vary between -1 and 1 to the 0-1 scale (see standard01). The default is FALSE.
thresh	argument to pass to threshMeasures if any of 'measures' is calculated by that function. The default is NULL, but a valid method must be specified if any of 'measures' is threshold-based - i.e., any of those in 'modEvAmethods("threshMeasures")'.
bin.method	the method with which to divide the data into groups or bins, for calibration or reliability measures such as HLfit. The default is NULL, but a valid method must be specified if 'measures' includes "HL" or "HL.p". Type modEvAmethods("getBins") for available options), and see HLfit and getBins for more information.

verbosity	integer specifying the amount of messages or warnings to display. Defaults to 0, but can also be 1 or 2 for more messages from the functions within.
...	optional arguments to pass to HLfit (if "HL" or "HL.p" are included in 'measures'), namely n.bins, fixed.bin.size, min.bin.size, min.prob.interval or quantile.type.

Value

A data frame with the value of each evaluation measure for each model.

Author(s)

A. Marcia Barbosa

See Also

threshMeasures

Examples

```
data(rotif.mods)

eval1 <- multModEv(models = rotif.mods$models[1:6], thresh = 0.5,
bin.method = "n.bins", fixed.bin.size = TRUE)

head(eval1)


eval2 <- multModEv(models = rotif.mods$models[1:6],
thresh = "preval", measures = c("AUC", "AUCPR", "CCR",
"Sensitivity", "TSS"))

head(eval2)


# you can also calculate evaluation measures for a set of
# observed vs predicted data, rather than from model objects:

obses <- sapply(rotif.mods$models, `[[`, "y")
preds <- sapply(rotif.mods$models, `[[`, "fitted.values")

eval3 <- multModEv(obs.data = obses[ , 1:4],
pred.data = preds[ , 1:4], thresh = "preval",
bin.method = "prob.bins")

head(eval3)
```

OA *Overlap Analysis*

Description

This function analyses the range of values of the given environmental variables at the sites where a species has been recorded present.

Usage

```
OA(data, sp.cols, var.cols)
```

Arguments

`data`	a data frame with your species' occurrence data and the predictor variables.
`sp.cols`	index number of the column containing the occurrence data of the species to be modelled. Currently only one species can be analysed at a time.
`var.cols`	index numbers of the columns containing the predictor variables to be used.

Details

Overlap Analysis is one of the simplest forms of modelling species' distributions. It assesses the ranges of values of the given environmental variables at the sites where a species has been recorded present, and predicts where that species should be able to occur based on those presence data (e.g. Brito et al. 1999, Arntzen & Teixeira 2006).

OA can also be useful when extrapolating models outside their original scope (geographical area, time period or spatial resolution), as it can identify which localities are within the model's domain - i.e., within the analysed ranges of values of the variables, outside which the model may not be reliable (e.g. Barbosa et al. 2009). In this case, the response is not a species' presence, but rather the sites that have been included in the model. See also the `MESS` function for a comparison between modelled and extrapolation environments.

Input data for the `OA` function are a vector or column with ones and zeros (presences vs. absences of a species if we want to model its occurrence, or modelled vs. non-modelled sites if we want to know which non-modelled sites are within the modelled range), and a matrix or data frame with the corresponding values of the environmental variables to consider (one variable in each column, values in rows).

Value

A binary vector whith 1 where the values of all predictors lie within the ranges observed for the presence records, and 0 otherwise.

Author(s)

A. Marcia Barbosa

References

Arntzen J.W, Teixeira J. (2006) History and new developments in the mapping and modelling of the distribution of the golden-striped salamander, Chioglossa lusitanica. Zeitschrift fur Feldherpetologie, Supplement: 1-14.

Barbosa, A.M., Real, R. & Vargas, J.M. (2009) Transferability of environmental favourability models in geographic space: the case of the Iberian desman (Galemys pyrenaicus) in Portugal and Spain. Ecological Modelling 220: 747-754.

Brito J.C., Crespo E.G., Paulo O.S. (1999) Modelling wildlife distributions: Logistic Multiple Regression vs Overlap Analysis. Ecography 22: 251-260.

See Also

MESS

Examples

```
## Not run:
# load package 'fuzzySim' and its sample data:
require(fuzzySim)
data(rotif.env)

names(rotif.env)

OA(rotif.env, sp.cols = 18, var.cols = 5:17)

## End(Not run)
```

`optiPair` *Optimize the classification threshold for a pair of related model evaluation measures.*

Description

This function can optimize a model's classification threshold based on a pair of model evaluation measures that balance each other, such as sensitivity-specificity, precision-recall (i.e., positive predictive power vs. sensitivity), or omission-commission, or underprediction-overprediction (Fielding & Bell 1997; Liu et al. 2011; Barbosa et al. 2013). The function plots both measures of the given pair against all thresholds with a given interval, and calculates the optimal sum, difference and mean of the two measures.

Usage

```
optiPair(model = NULL, obs = NULL, pred = NULL,
measures = c("Sensitivity", "Specificity"), interval = 0.01, pbg = FALSE,
plot = TRUE, plot.sum = FALSE, plot.diff = FALSE, col1 = "steelblue4",
col2 = "lightblue3", ylim = NULL, na.rm = TRUE, exclude.zeros = TRUE,
rm.dup = FALSE, verbosity = 2, ...)
```

Arguments

model	optional binary-response model object of class "glm", "gam", "gbm", "randomForest" or "bart". If this argument is provided, 'obs' and 'pred' will be extracted with mod2obspred. Alternatively, you can input the 'obs' and 'pred' arguments instead of 'model'.
obs	alternatively to 'model' and together with 'pred', a numeric vector of observed presences (1) and absences (0) of a binary response variable. Alternatively (and if 'pred' is a 'SpatRaster'), a two-column matrix or data frame containing, respectively, the x (longitude) and y (latitude) coordinates of the presence points, in which case the 'obs' vector will be extracted with ptsrast2obspred. This argument is ignored if 'model' is provided.
pred	alternatively to 'model' and together with 'obs', a vector with the corresponding predicted values of presence probability, habitat suitability, environmental favourability or alike. Must be of the same length and in the same order as 'obs'. Alternatively (and if 'obs' is a set of point coordinates), a 'SpatRaster' map of the predicted values for the entire evaluation region, in which case the 'pred' vector will be extracted with ptsrast2obspred. This argument is ignored if 'model' is provided.
measures	a character vector of length 2 indicating the pair of measures whose curves to plot and whose combined threshold to optimize. Available measures can be obtained with 'modEvAmethods("threshMeasures")', but note that this function expects you to use two measures that counter-balance one another, such as c("Sensitivity", "Specificity") [the default], c("Omission", "Commission"), or c("Precision", "Recall").
interval	the interval of thresholds at which to calculate the measures. The default is 0.01.
pbg	logical value to pass to inputMunch indicating whether to use presence/background (rather than presence/absence) data. Default FALSE.
plot	logical indicating whether or not to plot the pair of measures.
plot.sum	logical, whether to plot the sum (+) of both measures in the pair. Defaults to FALSE.
plot.diff	logical, whether to plot the difference (-) between both measures in the pair. Defaults to FALSE.
col1	plot colour for the 1st element in 'measures'.
col2	plot colour for the 2nd element in 'measures'.
ylim	a character vector of length 2 indicating the lower and upper limits for the y axis. The default is NULL for an automatic definition of 'ylim' based on the values of the measures and their sum and/or difference if any of these are set to TRUE.
na.rm	logical, whether NA values should be removed from the calculation of minimum/maximum/mean values to get the optimized measures. Defaults to TRUE.
exclude.zeros	logical, whether non-finite and zero values should be removed from the calculation of minimum/maximum/mean values to get the optimized measures. Defaults to TRUE.
rm.dup	If TRUE and if 'pred' is a SpatRaster and if there are repeated points within the same pixel, a maximum of one point per pixel is used to compute the presences. See examples in ptsrast2obspred. The default is FALSE.

verbosity	integer specifying the amount of messages to display. Defaults to the maximum implemented; lower numbers (down to 0) decrease the number of messages.
...	additional arguments to be passed to the plot function.

Value

The output is a list with the following components:

measures.values	a data frame with the values of the chosen pair of measures, as well as their difference, sum and mean, at each threshold.
MinDiff	numeric value, the minimum difference between both measures.
ThreshDiff	numeric value, the threshold that minimizes the difference between both measures.
MaxSum	numeric value, the maximum sum of both measures.
ThreshSum	numeric value, the threshold that maximizes the sum of both measures.
MaxMean	numeric value, the maximum mean of both measures.
ThreshMean	numeric value, the threshold that maximizes the mean of both measures.

If plot=TRUE (the default), a plot is also produced with the value of each of 'measures' at each threshold, and horizontal and vertical lines marking, respectively, the threshold and value at which the difference between the two 'measures' is minimal.

Author(s)

A. Marcia Barbosa

References

Barbosa, A.M., Real, R., Munoz, A.-R. & Brown, J.A. (2013) New measures for assessing model equilibrium and prediction mismatch in species distribution models. Diversity and Distributions 19: 1333-1338

Fielding A.H. & Bell J.F. (1997) A review of methods for the assessment of prediction errors in conservation presence/absence models. Environmental Conservation 24: 38-49

Liu C., White M., & Newell G. (2011) Measuring and comparing the accuracy of species distribution models with presence-absence data. Ecography, 34, 232-243.

See Also

getThreshold, optiThresh, threshMeasures

Examples

```
# load sample models:
data(rotif.mods)


# choose a particular model to play with:
```

```
mod <- rotif.mods$models[[1]]

optiPair(model = mod)

optiPair(model = mod, measures = c("Precision", "Recall"))

optiPair(model = mod, measures = c("UPR", "OPR"))

optiPair(model = mod, measures = c("CCR", "F1score"))


# you can also use 'optiPair' with vectors of observed
# and predicted values, instead of a model object:

optiPair(obs = mod$y, pred = mod$fitted.values)


# 'obs' can also be a table of presence point coordinates
# and 'pred' a SpatRaster of predicted values
```

`optiThresh` *Optimize threshold for model evaluation.*

Description

This function computes a number of model evaluation measures at optimal thresholds (see `threshMeasures`). Optimization is given for each measure, and/or for all measures according to particular criteria (e.g. Jimenez-Valverde & Lobo 2007; Liu et al. 2005; Nenzen & Araujo 2011). Results are given numerically and in plots. See `getThreshold` if you're after the optimized threshold rather than the threshold-optimized evaluation measures.

Usage

```
optiThresh(model = NULL, obs = NULL, pred = NULL, interval = 0.01,
measures = c(modEvAmethods("threshMeasures"), modEvAmethods("similarity")),
optimize = modEvAmethods("optiThresh"), simplif = FALSE, pbg = FALSE,
plot = TRUE, sep.plots = FALSE, reset.par = TRUE, xlab = "Threshold",
na.rm = TRUE, rm.dup = FALSE, pch = 20, cex = 0.2, col = "steelblue4",
verbosity = 2, ...)
```

Arguments

`model`	optional binary-response model object of class "glm", "gam", "gbm", "randomForest" or "bart". If this argument is provided, 'obs' and 'pred' will be extracted with `mod2obspred`. Alternatively, you can input the 'obs' and 'pred' arguments instead of 'model'.

obs	alternatively to 'model' and together with 'pred', a numeric vector of observed presences (1) and absences (0) of a binary response variable. Alternatively (and if 'pred' is a 'SpatRaster'), a two-column matrix or data frame containing, respectively, the x (longitude) and y (latitude) coordinates of the presence points, in which case the 'obs' vector will be extracted with ptsrast2obspred. This argument is ignored if 'model' is provided.
pred	alternatively to 'model' and together with 'obs', a vector with the corresponding predicted values of presence probability, habitat suitability, environmental favourability or alike. Must be of the same length and in the same order as 'obs'. Alternatively (and if 'obs' is a set of point coordinates), a 'SpatRaster' map of the predicted values for the entire evaluation region, in which case the 'pred' vector will be extracted with ptsrast2obspred. This argument is ignored if 'model' is provided.
pbg	logical value to pass to inputMunch indicating whether to use presence/background (rather than presence/absence) data. Default FALSE.
interval	numeric value between 0 and 1 indicating the interval between the thresholds at which to calculate the evaluation measures. Defaults to 0.01.
measures	character vector indicating the names of the model evaluation measures for which to calculate optimal thresholds. The default is using all measures available in 'c(modEvAmethods("threshMeasures"), modEvAmethods("similarity"))'.
optimize	character vector indicating the threshold optimization criteria to use; "each" calculates the optimal threshold for each model evaluation measure, while the remaining options optimize all measures according to the specified criterion. The default is using all criteria available in 'modEvAmethods("optiThresh")'.
simplif	logical, whether to compute a faster simplified version. Used internally in other functions.
plot	logical, whether to plot the values of each evaluation measure at all thresholds. Ignored if simplif=TRUE.
sep.plots	logical. If TRUE, each plot is shown separately (you need to be recording R plot history to be able to browse through them all); if FALSE (the default), plots are combined (using arrangePlots) in the same plotting window; if NA, the current par()["mfrow"] is used.
reset.par	logical. If TRUE (the default), plotting parameters are reset in the end. FALSE can be useful if the user wants to set their own parameters and combine this with other plots.
xlab	character vector indicating the label of the x axis.
na.rm	Logical value indicating whether missing values should be ignored in computations. Defaults to TRUE.
rm.dup	If TRUE and if 'pred' is a SpatRaster and if there are repeated points within the same pixel, a maximum of one point per pixel is used to compute the presences. See examples in ptsrast2obspred. The default is FALSE.
pch	plotting character (symbol), argument to pass to plot.
cex	character expansion (size), argument to pass to plot.
col	colour, argument to pass to plot.

`verbosity` integer specifying the amount of messages to display. Defaults to the maximum implemented; lower numbers (down to 0) decrease the number of messages.

`...` additional arguments that can be passed to `plot`.

Value

This function returns a list with the following components:

`all.thresholds` a data frame with the values of all analysed measures at all analysed thresholds.

`optimals.each` if "each" is among the threshold criteria specified in 'optimize', optimals.each is output as a data frame with the value of each measure at its optimal threshold, as well as the type of optimal for that measure (which may be the maximum for measures of goodness such as "Sensitivity", or the minimum for measures of badness such as "Omission").

`optimals.criteria` a data frame with the values of measure at the threshold that maximizes each of the criteria specified in 'optimize' (except for "each", see above).

Note

"Sensitivity" is the same as "Recall", and "PPP" (positive predictive power) is the same as "Precision". "F1score" is the harmonic mean of precision and recall.

Note

Some measures cannot be calculated for thresholds at which there are zeros in the confusion matrix, hence the eventual 'NaN' or 'Inf' in results. Also, optimization may be deceiving for some measures; use 'plot = TRUE' and inspect the plot(s).

Author(s)

A. Marcia Barbosa

References

Jimenez-Valverde A. & Lobo J.M. (2007) Threshold criteria for conversion of probability of species presence to either-or presence-absence. Acta Oecologica 31: 361-369.

Liu C., Berry P.M., Dawson T.P. & Pearson R.G. (2005) Selecting thresholds of occurrence in the prediction of species distributions. Ecography 28: 385-393.

Nenzen H.K. & Araujo M.B. (2011) Choice of threshold alters projections of species range shifts under climate change. Ecological Modelling 222: 3346-3354.

See Also

`getThreshold`, `optiPair`, `threshMeasures`

Examples

```
# load sample models:
data(rotif.mods)

# choose a particular model to play with:
mod <- rotif.mods$models[[1]]

## Not run:
optiThresh(model = mod)

## End(Not run)


# change some of the parameters:

optiThresh(model = mod, pch = 20,
measures = c("CCR", "Sensitivity", "kappa", "TSS", "Jaccard", "F1score"),
ylim = c(0, 1))


# you can also use optiThresh with vectors of observed and predicted
# values instead of with a model object:

## Not run:
optiThresh(obs = mod$y, pred = mod$fitted.values, pch = 20)

## End(Not run)


# 'obs' can also be a table of presence point coordinates
# and 'pred' a SpatRaster of predicted values
```

`plotCoeffs` *Plot model coefficients with confidence intervals*

Description

Gets the coefficient estimates of a model with their 95% confidence intervals, and shows them in a point range plot.

Usage

```
plotCoeffs(model, labels = NULL, plot = TRUE, ...)
```

Arguments

`model`	A model object whose summary() coefficients table has column names matching (case-insensitive) "estimate", "err" and "P" - e.g., of class "glm", "lm", "phylolm" or "glmerMod".

labels	Optional two-column matrix or data frame, with the first column containing names of variables included in 'model' (in no particular order), and the second column containing the corresponding names to use as labels in the plot. Can be useful when the model contains abbreviated variable names, and you want more complete variable names in the plot.
plot	Logical value. If set to FALSE, only the results data frame is produced, and no plot.
...	(If plot=TRUE) additional arguments to pass to lollipop, e.g. horizontal = TRUE.

Details

This function uses summary to get the coefficients (excluding the intercept) and their standard errors from a fitted model object, and it computes the 95% confidence intervals, i.e. each coefficient plus/minus 1.96 times its standard error. By default, it also uses lollipop to produce a point range plot with those results. In the plot, the labels of coefficients with significant p-values (<= 0.05) are displayed in bold.

Note that coefficients are only directly comparable if the variables were standardized or if they vary in the same scale.

Value

A data frame (and a plot).

Author(s)

A. Marcia Barbosa

See Also

varImp, lollipop, confint

Examples

```
# get example model:

mod <- glm(am ~ mpg + wt + gear + carb, data = as.data.frame(scale(mtcars)))

summary(mod)


# default plot:

plotCoeffs(mod)


# with some plot tweaks:

labels <- data.frame(
  vars = c("gear", "wt", "mpg", "carb"),
```

```
  labs = c("N gears", "Weight", "Miles per gallon", "N carburators")
  )

par(mar = c(3, 10, 2, 1))

plotCoeffs(mod, horiz = TRUE, cex = 2, grid = FALSE, axis.lab = "Coefficients",
main = "Transmission", labels = labels, col = "darkturquoise")
```

`plotGLM` *Plot a generalized linear model*

Description

This function plots the observed (presence/absence) data and the predicted (probability) values of a Generalized Linear Model against the y regression equation (logit) values. Only logistic regression (binomial response, logit link) is currently implemented.

Usage

```
plotGLM(model = NULL, obs = NULL, pred = NULL, link = "logit",
plot.values = TRUE, plot.digits = 3, xlab = "Logit (Y)",
ylab = "Predicted probability", main = "Model plot", na.rm = TRUE,
rm.dup = FALSE, verbosity = 2, ...)
```

Arguments

`model`	optional binary-response model object of class "`glm`". If this argument is provided, 'obs' and 'pred' will be extracted with `mod2obspred`. Alternatively, you can input the 'obs' and 'pred' arguments instead of 'model'.
`obs`	alternatively to 'model' and together with 'pred', a numeric vector of observed presences (1) and absences (0) of a binary response variable. Alternatively (and if 'pred' is a 'SpatRaster'), a two-column matrix or data frame containing, respectively, the x (longitude) and y (latitude) coordinates of the presence points, in which case the 'obs' vector will be extracted with `ptsrast2obspred`. This argument is ignored if 'model' is provided.
`pred`	alternatively to 'model' and together with 'obs', a numeric vector with the corresponding predicted values of presence probability, habitat suitability, environmental favourability or alike. Must be of the same length and in the same order as 'obs'. Alternatively (and if 'obs' is a set of point coordinates), a 'SpatRaster' map of the predicted values for the entire evaluation region, in which case the 'pred' vector will be extracted with `ptsrast2obspred`. This argument is ignored if 'model' is provided.
`link`	the link function of the GLM; only 'logit' (the default) is implemented.
`plot.values`	logical, whether to include in the plot diagnostic values such as explained deviance (calculated with the `Dsquared` function) and pseudo-R-squared measures (calculated with the `pseudoRsq` function). Defaults to TRUE.

plot.digits	integer number indicating the number of digits to which the values in the plot should be rounded (if 'plot.values = TRUE'). Defaults to 3.
xlab	character string specifying the label for the x axis.
ylab	character string specifying the label for the y axis.
main	character string specifying the title for the plot.
na.rm	Logical value indicating whether missing values should be ignored in computations. Defaults to TRUE.
rm.dup	If TRUE and if 'pred' is a SpatRaster and if there are repeated points within the same pixel, a maximum of one point per pixel is used to compute the presences. See examples in ptsrast2obspred. The default is FALSE.
verbosity	integer specifying the amount of messages to display. Defaults to the maximum implemented; lower numbers (down to 0) decrease the number of messages.
...	additional arguments to pass to plot.

Value

This function outputs a plot of model predictions against observations.

Author(s)

A. Marcia Barbosa

References

Guisan A. & Zimmermann N.E. (2000) Predictive habitat distribution models in ecology. Ecological Modelling 135: 147-186

Weisberg S. (1980) Applied Linear Regression. Wiley, New York

See Also

predPlot, predDensity

Examples

```
# load sample models:
data(rotif.mods)

# choose a particular model to play with:
mod <- rotif.mods$models[[1]]

plotGLM(model = mod)

plotGLM(model = mod, plot.values = FALSE)


# you can also use 'plotGLM' with vectors of observed and
# predicted values instead of with a model object:
```

```
plotGLM(obs = mod$y, pred = mod$fitted.values)


# 'obs' can also be a table of presence point coordinates
# and 'pred' a SpatRaster of predicted values
```

---

poMeasures *Strictly presence-only measures of model evaluation*

Description

THIS FUNCTION IS EXPERIMENTAL AND IMPLEMENTS METHODS THAT ARE STILL UNDER SCIENTIFIC REVIEW. It computes a set of truly presence-only evaluation metrics for model predictions, including bin-dependent and bin-independent approaches. The function evaluates how the continuous predicted values (at presence sites only) relate to the number or density of observed presences, using correlation-based measures and distributional comparisons. Optional diagnostic plots are also produced.

Usage

```
poMeasures(
  model = NULL,
  obs = NULL,
  pred = NULL,
  n.bins = 100,
  bin.width = "default",
  method = "spearman",
  rm.dup.classes = NA,
  rm.dup.points = FALSE,
  simplif = FALSE,
  plot = TRUE,
  plot.lines = TRUE,
  plot.values = TRUE,
  cex.values = 1,
  plot.digits = 3,
  main = c("Presence-only bin-based", "Presence-only density-based"),
  na.rm = TRUE,
  verbosity = 2,
  ...
)
```

Arguments

model	optional binary-response model object of class "glm", "gam", "gbm", "randomForest" or "bart", producing prediciotns between 0 and 1. If this argument is provided, 'obs' and 'pred' will be extracted with mod2obspred. Alternatively, you can input the 'obs' and 'pred' arguments (e.g. for external test data) instead of 'model'.

obs	alternatively to 'model' and (optionally) together with 'pred', a numeric vector of observed presences (1) and absences (0) of a binary response variable. Alternatively (and if 'pred' is a 'SpatRaster'), a two-column matrix or data frame containing, respectively, the x (longitude) and y (latitude) coordinates of the presence points, in which case the 'obs' vector will be extracted with ptsrast2obspred. This argument is ignored if 'model' is provided. If used, rows where obs = 0 or pixels with no presence points are removed before calculations.
pred	alternatively to 'model' and (optionally) together with 'obs', a vector with the corresponding predicted values of habitat suitability, environmental favourability or alike (presence probability is not appropriate for these measures, unless model training prevalence equals 0.5). Values should lie between 0 and 1. If 'obs' is provided, 'pred' must be of the same length and in the same order. Alternatively (and if 'obs' is a set of point coordinates), a 'SpatRaster' map of the predicted values (at presence sites only, unless 'obs' coordinates are provided), in which case the 'pred' vector will be extracted with ptsrast2obspred. Values should lie between 0 and 1. This argument is ignored if 'model' is provided.
n.bins	argument to pass to getBins indicating the number of bins in which to divide the data.
bin.width	argument to pass to getBins indicating the width of the moving window. The default is 1/10th of the 'pred' range.
method	argument to pass to cor.test specifying the correlation method to use for POGI. The default is "spearman", as in the initial publication on the Boyce index on which these measures are inspired, but "kendall" or "pearson" could be more informative in many cases.
rm.dup.classes	(placeholder, as in Boyce, but currently not used here).
rm.dup.points	if TRUE and if 'pred' is a SpatRaster and if there are >1 points within the same pixel, a maximum of one point per pixel is used to compute the presences. See examples in ptsrast2obspred. The default is FALSE.
simplif	logical value (default FALSE) indicating whether to return just a simplified output containing only the value estimate for each measure, with no associated statistics.
plot	Logical; if TRUE (default), plot the values on which the measures are based.
plot.lines	(if plot=TRUE) logical value (default TRUE) indicating whether or not to add lines connecting the points in the moving bins plot. Missing value (NA) bins are not connected (see lines).
plot.values	logical (default TRUE), whether or not to show the measure values on the plots.
cex.values	Character expansion (size) for plot values text.
plot.digits	(if plot=TRUE and plot.values=TRUE) number of digits (default 3) to which the values in the plots should be rounded.
main	(if plot=TRUE) main titles for the two plots.
na.rm	Logical (default TRUE) indicating if missing values should be removed from computations.

verbosity	integer specifying the amount of console messages to display. Defaults to the maximum implemented; lower numbers (down to 0) decrease the number of messages.
...	(if plot=TRUE) additional graphical parameters that can be passed to plot().

Value

A list with the following components:

bins	A data frame containing bin statistics from getBins.
Nbins	Number of bins with (not missing, non-NA) data.
bPOGI	bin-based Presence-Only Gradient Index.
dPOGI	density-based (bin-independent) Presence-Only Gradient Index.
PODS	Presence-Only Distributional Shift (PODS).
N	Number of predicted values (at presence sites if 'obs' provided).

If plot = TRUE, two diagnostic plots are also produced.

See Also

getBins, Boyce, cor.test, wilcox.test

Examples

```
# load sample models:
data(rotif.mods)

# choose a particular model to play with:
mod <- rotif.mods$models[[18]]

# compute the PO measures:
# poMeasures(model = mod)


# you can also use vectors of observed and predicted values
# instead of a model object:

obs <- mod$y
pred <- mod$fitted.values

poMeasures(obs = obs, pred = pred)


# 'obs' can also be a table of presence point coordinates
# and 'pred' a SpatRaster of predicted values
```

predDensity *Plot the density of predicted or predictor values for presences and absences (or background)*

Description

This function produces a histogram and/or a kernel density plot of predicted or predictor values for a binary-response model, optionally separately for the observed presences and absences (or background, if pbg = TRUE), given either a model object or a vector of predicted (or predictor) values, and optionally a vector of the corresponding observed values (with 0 for absence and 1 for presence). When there are multiple predicted values for each site (e.g. for BART models), it can also plot a confidence interval.

Usage

```
predDensity(model = NULL, obs = NULL, pred = NULL, separate = TRUE,
type = "both", ci = NA, pbg = FALSE, bw = "nrd0", legend.pos = "topright",
main = "Density of pred values", na.rm = TRUE, rm.dup = FALSE, xlim = NULL,
verbosity = 2, ...)
```

Arguments

model	optional binary-response model object of class "glm", "gam", "gbm", "randomForest" or "bart". If this argument is provided, 'obs' and 'pred' will be extracted with mod2obspred. Alternatively, you can input the 'pred' (and optionally 'obs') argument(s) instead of 'model'.
obs	alternatively to 'model' and together with 'pred', an optional numeric vector (in the same order of 'pred') of observed presences (1) and absences (0) of a binary response variable. If pbg = TRUE, the presences will be appended to the absences to create the background. Alternatively (and if 'pred' is a 'SpatRaster'), a two-column matrix or data frame containing, respectively, the x (longitude) and y (latitude) coordinates of the presence points, in which case the 'obs' vector will be extracted with ptsrast2obspred. This argument may be omitted (to show the density plot of all 'pred' values combined), and it is ignored if 'model' is provided.
pred	alternatively to 'model', a vector of predicted values of presence probability, habitat suitability, environmental favourability or alike; or a vector of values of a continuous predictor variable. Must be of the same length and in the same order as 'obs' (if the latter is provided). Alternatively (and if 'obs' is a set of point coordinates), a 'SpatRaster' map of the predicted values for the entire evaluation region, in which case the 'pred' vector will be extracted with ptsrast2obspred. This argument is ignored if 'model' is provided.
separate	logical value indicating whether prediction densities should be computed separately for observed presences (ones) and absences (zeros), or for presences (ones) and background (ones and zeros) if pbg = TRUE. Defaults to TRUE, but it is automatically changed to FALSE if either 'model' or 'obs' are not provided, or if 'ci' is not NA.

type	character vector specifying whether to produce a "histogram", a "density" plot, or "both" (the default). Partial argument matching is used.
ci	optional numeric value for a confidence interval to add to the plot, e.g. 0.95 for 95%. The default is NA. If specified, argument 'separate' is set to FALSE.
pbg	logical value (default FALSE) to pass to inputMunch indicating whether to use presence/background (rather than presence/absence) data.
bw	argument to pass to density indicating the the smoothing bandwidth to use.
legend.pos	character specifying the position for the legend; NA or "n" for no legend. Position can be "topright" (the default), "topleft, "bottomright"", "bottomleft", "top", "bottom", "left", "right", or "center". Partial argument matching is used.
main	main title for the plot.
na.rm	logical value indicating whether missing values should be ignored in computations. Defaults to TRUE.
rm.dup	if TRUE and if 'pred' is a SpatRaster and if there are repeated points within the same pixel, a maximum of one point per pixel is used to compute the presences. See examples in ptsrast2obspred. The default is FALSE.
xlim	numeric vector of length 2 setting the limits for the x axis of the plot. The default is NULL, for the range of the density of predicted or predictor values.
verbosity	integer specifying the amount of messages to display. Defaults to the maximum implemented; lower numbers (down to 0) decrease the number of messages.
...	additional arguments to pass to hist, e.g. 'breaks' or 'border'.

Value

This function outputs and plots a list containing the object(s) specified in 'type' - by default, a density object and a histogram of the input predicted or predictor values, optionally (if separate=TRUE) separately for presence and absence rows. If separate=TRUE, the output also includes the results of a wilcox.test and a ks.test comparing those values.

Author(s)

A. Marcia Barbosa

See Also

predPlot

Examples

```
# load sample models:
data(rotif.mods)

# choose a particular model to play with:
mod <- rotif.mods$models[[1]]


# compute predDensity with different parameters:
```

```
predDensity(model = mod)

predDensity(model = mod, breaks = seq(0, 1, by = 0.05))

predDensity(model = mod, type = "histogram")

predDensity(model = mod, type = "density")

predDensity(model = mod, ci = 0.975)


# you can also use 'predDensity' with vectors of
# observed and predicted values, instead of a model object:

obs <- mod$y
pred <- mod$fitted.values

predDensity(obs = obs, pred = pred)

predDensity(pred = pred, ci = 0.95)


# 'obs' can also be a table of presence point coordinates
# and 'pred' a SpatRaster of predicted values
```

`predPlot` *Plot predicted values for presences and absences, optionally classified according to a prediction threshold.*

Description

This function plots predicted values separated into observed presences and absences and (optionally and by default) coloured according to whether they are above or below a given prediction threshold. The plot imitates (with permission from the author) one of the graphical outputs of the 'summary' of models built with the **embarcadero** package (Carlson, 2020), but it can be applied to other types of models or to a set of observed and predicted values, and it allows specifying a user-defined threshold.

Usage

```
predPlot(model = NULL, obs = NULL, pred = NULL, thresh = "preval", pbg = FALSE,
main = "Classified predicted values", legend.pos = "n", pch = 1, cex = 0.5,
col = c("steelblue4", "lightblue2"), na.rm = TRUE, rm.dup = FALSE,
interval = 0.01, quant = 0, verbosity = 2)
```

Arguments

model	optional binary-response model object of class "glm", "gam", "gbm", "randomForest" or "bart". If this argument is provided, 'obs' and 'pred' will be extracted with mod2obspred. Alternatively, you can input the 'obs' and 'pred' arguments instead of 'model'.
obs	alternatively to 'model' and together with 'pred', a numeric vector of observed presences (1) and absences (0) of a binary response variable. Alternatively (and if 'pred' is a 'SpatRaster'), a two-column matrix or data frame containing, respectively, the x (longitude) and y (latitude) coordinates of the presence points, in which case the 'obs' vector will be extracted with ptsrast2obspred. This argument is ignored if 'model' is provided.
pred	alternatively to 'model' and together with 'obs', a vector with the corresponding predicted values of presence probability, habitat suitability, environmental favourability or alike. Must be of the same length and in the same order as 'obs'. Alternatively (and if 'obs' is a set of point coordinates), a 'SpatRaster' map of the predicted values for the entire evaluation region, in which case the 'pred' vector will be extracted with ptsrast2obspred. This argument is ignored if 'model' is provided.
thresh	threshold value to separate predicted presences from predicted absences in 'pred'; can be "preval" (the default), to use the prevalence (i.e. proportion of presences) in 'obs'; or any real number between 0 and 1; or any of the options available on modEvAmethods("getThreshold") – see Details in getThreshold for their description. This value, if not NA or NULL, will be used to draw a vertical line on the plot and to colour the points (predicted values) according to whether they fall above or below the threshold.
pbg	logical value to pass to inputMunch indicating whether to use presence/background (rather than presence/absence) data. Default FALSE.
main	Main title for the plot.
legend.pos	character value specifying the position for the legend on the plot. Can be "bottomleft", "bottom", "bottomright", "topleft", "left", "top", "topright", "right", "center", or NA or "n" for no legend (the default). Partial argument matching is used.
pch	plotting character for the presences and absences (see par).
cex	relative size of the plotting character (see par).
col	vector of length 2 indicating the colours with which to plot predicted presences and absences (points above and below the threshold), respectively. If 'thresh' is NA or NULL, all points will have the first of the specified colours.
na.rm	Logical value indicating whether missing values should be ignored in computations. Defaults to TRUE.
rm.dup	If TRUE and if 'pred' is a SpatRaster and if there are repeated points within the same pixel, a maximum of one point per pixel is used to compute the presences. See examples in ptsrast2obspred. The default is FALSE.
interval	Argument to pass to optiThresh indicating the interval between the thresholds to test, if 'thresh' implies optimizing a threshold-based measure. The default

is 0.01. Smaller values may provide more precise results but take longer to compute.

`quant` Numeric value indicating the proportion of presences to discard if thresh="MTP" (minimum training presence). With the default value 0, MTP will be the threshold at which all observed presences are classified as such; with e.g. quant=0.05, MTP will be the threshold at which 5% presences will be classified as absences.

`verbosity` integer specifying the amount of messages to display. Defaults to the maximum implemented; lower numbers (down to 0) decrease the number of messages.

Value

This function outputs a plot as per 'Description'.

Note

Points are `jitter`ed randomly along the y axis to minimize visual overlap. So, each run of 'predPlot' (unless you use `set.seed` first) will produce a different arrangement of points for the same data, although their x-axis values are faithful.

Author(s)

A. Marcia Barbosa

References

Carlson C.J. (2020) embarcadero: Species distribution modelling with Bayesian additive regression trees in R. Methods in Ecology and Evolution, 11: 850-858.

See Also

`predDensity`, `plotGLM`

Examples

```
# load sample models:
data(rotif.mods)

# choose a particular model to play with:
mod <- rotif.mods$models[[1]]

predPlot(model = mod)

predPlot(model = mod, thresh = 0.5)


# you can first select a threshold optimized according to a particular metric:

## Not run:
threshold <- optiThresh(mod, measures = "TSS", optimize = "each")
threshold <- threshold$optimals.each[ , "threshold"]
threshold
```

```
predPlot(model = mod, thresh = threshold)

## End(Not run)

# you can also use 'predPlot' with vectors of observed and predicted values
# instead of a model object:

presabs <- mod$y
prediction <- mod$fitted.values

predPlot(obs = presabs, pred = prediction)

predPlot(obs = presabs, pred = prediction, thresh = 0.5)


# 'obs' can also be a table of presence point coordinates
# and 'pred' a SpatRaster of predicted values
```

`prevalence` *Prevalence*

Description

For building and evaluating species distribution models, the porportion of presences of the species may be an issue to take into account (e.g. Jimenez-Valverde & Lobo 2006, Barbosa et al. 2013). The `prevalence` function calculates this measure.

Usage

```
prevalence(obs, model = NULL, event = 1, na.rm = TRUE)
```

Arguments

`obs`	a vector or a factor of binary observations (e.g. 1 vs. 0, male vs. female, disease vs. no disease, etc.). This argument is ignored if 'model' is provided.
`model`	alternatively to 'obs', a binary-response model object of class "glm", "gam", "gbm", "randomForest" or "bart". If this argument is provided, 'obs' will be extracted with `mod2obspred`.
`event`	the value whose prevalence we want to calculate (e.g. 1, "present", etc.). This argument is ignored if 'model' is provided.
`na.rm`	logical, whether NA values should be excluded from the calculation. The default is TRUE.

Value

Numeric value of the prevalence of `event` in the `obs` vector.

Author(s)

A. Marcia Barbosa

References

Barbosa A.M., Real R., Munoz A.R. & Brown J.A. (2013) New measures for assessing model equilibrium and prediction mismatch in species distribution models. Diversity and Distributions, in press

Jimenez-Valverde A. & Lobo J.M. (2006) The ghost of unbalanced species distribution data in geographical model predictions. Diversity and Distributions, 12: 521-524.

See Also

evenness

Examples

```
# calculate prevalence from binary vectors:

(x <- rep(c(0, 1), each = 5))

(y <- c(rep(0, 3), rep(1, 7)))

(z <- c(rep(0, 7), rep(1, 3)))

prevalence(x)

prevalence(y)

prevalence(z)


(w <- c(rep("yes", 3), rep("nope", 7)))

prevalence(w, event = "yes")


# calculate prevalence from a model object:

data(rotif.mods)

prevalence(mod = rotif.mods$models[[1]])
```

---

pseudoRsq *Pseudo-R-squared measures for binary-response models*

Description

This function computes several pseudo R-squared statistics for a binomial Generalized Linear Model object, or for observations and predictions of any binary-response model.

Usage

```
pseudoRsq(model = NULL, obs = NULL, pred = NULL, use = "pairwise.complete.obs",
plot = TRUE, plot.type = "lollipop", na.rm = TRUE, rm.dup = FALSE,
verbosity = 2, ...)
```

Arguments

model	optional binary-response model object of class "glm". Alternatively, you can input the 'obs' and 'pred' values of any other binary-response model.
obs	alternatively to 'model' and together with 'pred', a vector of observed presences (1) and absences (0) of a binary response variable. Alternatively (and if 'pred' is a 'SpatRaster'), a two-column matrix or data frame containing, respectively, the x (longitude) and y (latitude) coordinates of the presence points, in which case the 'obs' vector will be extracted with ptsrast2obspred. This argument is ignored if 'model' is provided.
pred	alternatively to 'model' and together with 'obs', a numeric vector with the corresponding predicted values of presence probability. Must be of the same length and in the same order as 'obs'. Alternatively (and if 'obs' is a set of point coordinates), a 'SpatRaster' map of the predicted values for the entire evaluation region, in which case the 'pred' vector will be extracted with ptsrast2obspred. This argument is ignored if 'model' is provided.
use	argument to pass to cor for handling mising values.
plot	logical value indicating whether or not to display a bar chart or (by default) a lollipop chart of the output measures.
plot.type	(if plot=TRUE) character value indicating the type of plot to produce. Can be "lollipop" (the default) or "barplot".
na.rm	Logical value (default TRUE) indicating whether missing values should be ignored in computations.
rm.dup	Logical, default FALSE. If TRUE and if 'pred' is a SpatRaster and if there are >1 points within the same pixel, a maximum of one point per pixel is used to compute the presences (see Examples in ptsrast2obspred).
verbosity	integer specifying the amount of messages to display. Defaults to the maximum implemented; lower numbers (down to 0) decrease the number of messages.
...	additional arguments to pass to the plotting function (see Examples).

Details

Implemented measures include the pseudo R-squared values of McFadden (1974), Cox & Snell (1989), Nagelkerke (1991, which corresponds to the corrected Cox-Snell, eliminating its upper bound), and Tjur (2009). See Allison (2014) for a brief review of these measures. Note that Tjur's R-squared can only be calculated for models with a binomial response variable; otherwise, NA will be returned.

Note also that pseudo R-squared values tend to be considerably lower than those of the R-squared for ordinary regression analysis, and they should not be judged by the same standards for a "good fit". For example, for McFadden's R-squared, values of 0.2 to 0.4 represent an excellent fit (McFadden, 1979).

Value

The function returns a named list of the calculated R-squared values.

Note

This function was previously named RsqGLM.

Author(s)

A. Marcia Barbosa

References

Allison P. (2014) Measures of fit for logistic regression. SAS Global Forum, Paper 1485-2014

Cox, D.R. & Snell E.J. (1989) The Analysis of Binary Data, 2nd ed. Chapman and Hall, London

McFadden, D. (1974) Conditional logit analysis of qualitative choice behavior. In: Zarembka P. (ed.) Frontiers in Economics. Academic Press, New York

McFadden, D. (1979) Quantitative Methods for Analyzing Travel Behaviour on Individuals: Some Recent Developments. Chapter 15 in Behavioural Travel Modelling. Edited by David Hensher and Peter Stopher.

Nagelkerke, N.J.D. (1991) A note on a general definition of the coefficient of determination. Biometrika, 78: 691-692

Tjur T. (2009) Coefficients of determination in logistic regression models - a new proposal: the coefficient of discrimination. The American Statistician, 63: 366-372.

See Also

Dsquared, AUC, threshMeasures, HLfit, Miller

Examples

```
# load sample models:
data(rotif.mods)

# choose a particular model to play with:
mod <- rotif.mods$models[[1]]

pseudoRsq(model = mod)


# you can also use pseudoRsq with vectors of observed and predicted values
# instead of a model object:

pseudoRsq(obs = mod$y, pred = mod$fitted.values)


# plotting arguments can be modified:

par(mar = c(6, 3, 2, 1))
```

```
pseudoRsq(obs = mod$y, pred = mod$fitted.values, col = "seagreen", border = NA,
ylim = c(0, 1), main = "Pseudo-R-squared values")
```

---

ptsrast2obspred *Observed and predicted values from presence points and a raster map.*

Description

This function takes presence points or coordinates and a raster map of model predictions, and it returns a data frame with two columns containing, respectively, the observed (presence or no presence) and the predicted value for each pixel. Duplicate points (i.e., points falling in the same pixel, whether or not they have the exact same coordinates) can be kept or removed.

Usage

```
ptsrast2obspred(pts, rst, rm.dup = FALSE, na.rm = FALSE, verbosity = 2)
```

Arguments

pts	a 'SpatVector' map of the presence points, or a two-column matrix or data frame containing their x (longitude) and y (latitude) coordinates, respectively.
rst	a one-layer 'SpatRaster' map of the model predictions, in the same CRS as 'pts'. If you have a raster map in another format, you can try to convert it with 'terra::rast()'
rm.dup	logical, whether repeated points within the same pixel should be removed. See Examples. The default is FALSE.
na.rm	logical, whether presence points with missing or non-finite values of 'rst' should be excluded from the output. The default is FALSE.
verbosity	integer value indicating the amount of messages to display. Defaults to 2, for the maximum amount of messages.

Value

This function outputs a data frame with one column containing the observed (1 for presence, 0 for absence) and another column containing the corresponding predicted values from 'rst'.

Author(s)

A. Marcia Barbosa

Examples

```
## Not run:
# you can run these examples if you have the 'terra' package installed

require(terra)

# get an example raster map:
rst <- terra::rast(system.file("ex/elev.tif", package = "terra"))
rst <- terra::aggregate(rst, 10)

plot(rst)

# generate some random presence points within it:
set.seed(8)
presences <- terra::spatSample(as.polygons(ext(rst)), 10)

plot(presences, add = TRUE)


# use 'ptsrast2obspred' on this points + raster data:

# without removing duplicates (the default):
obspred <- ptsrast2obspred(pts = crds(presences), rst = rst)
obspred
nrow(obspred)  # you get as many 'obs' as pixels + additional points per pixel
sum(obspred$obs)  # as many presences as points that overlay 'pred'

# with removal of duplicates:
obspred_rmdup <- ptsrast2obspred(pts = crds(presences), rst = rst[[1]],
rm.dup = TRUE)  # you get as many 'obs' as pixels
obspred_rmdup
nrow(obspred_rmdup)  # you get as many 'obs' as pixels
sum(obspred_rmdup$obs)  # as many presences as pixels that contain (one or more) points

## End(Not run)
```

---

quantReclass *Reclassify continuous values based on quantiles*

Description

This function takes the continuous predictions of a model of suitability (e.g. the continuous Bioclim envelope model, computed by the `bioclim` function of the **dismo** package or the `envelope` function of the **predicts** package), and reclassifies them according to their quantiles.

Usage

```
quantReclass(pred, by = 0.01, na.rm = TRUE)
```

Arguments

pred	a 'numeric' vector or a 'SpatRaster' map of predicted suitability values.
by	numeric value indicating which quantiles to compute, e.g. 0.01 for percentiles (the default), 0.1 for deciles, etc.
na.rm	logical value indicating whether NA values should be ignored when computing the quantiles. Defaults to TRUE.

Details

This function was created by Formoso-Freire et al. (2023) to reclassify continuous Bioclim predictions into ranked suitability values, rescaling them into relative suitability. Modern implementations of Bioclim compute a percentile distribution of the values of each environmental variable at species presence localities. Then, the closer to the 50th percentile (the median), the more suitable a location is according to that variable (Hijmans et al. 2020; Hijmans 2023). However, the more variables are included in the model, the less suitable any location becomes, because it is less likely to be close to the median for all variables. The proposed rescaling procedure removes the dependence of Bioclim predictions on the number of variables included, and it has shown to provide more realistic predictions (Formoso-Freire et al., 2023).

Value

This function returns an object of the same class as 'pred' with the reclassified values.

Author(s)

A. Marcia Barbosa, Victoria Formoso-Freire, Andres Baselga, Carola Gomez-Rodriguez

References

Formoso-Freire V., Barbosa A.M., Baselga A., Gomez-Rodriguez C. (2023) Predicting the spatio-temporal pattern of range expansion under lack of equilibrium with climate. Biological Conservation, 288: 110361

Hijmans R.J., Phillips S., Leathwick J. & Elith J. (2020). dismo: Species distribution modelling (1.3.5). https://CRAN.R-project.org/package=dismo

Hijmans R.J. (2023). predicts: Spatial Prediction Tools. R package version 0.1-11. https://CRAN.R-project.org/package=predicts

See Also

getThreshold, bioclim in package **dismo**, envelope in package **predicts**

Examples

```
# simulate some sample data:
set.seed(2023)
bioclim_pred <- runif(n = 10, min = 0, max = 1)
bioclim_pred

quantReclass(pred = bioclim_pred, by = 0.1)
```

range01 *Shrink or stretch a vector to make it range between 0 and 1*

Description

This function re-scales a numeric vector so that it ranges between 0 and 1. So, the lowest value becomes 0, the highest becomes 1, and the ones in the middle retain their rank and relative diference.

Usage

```
range01(x, na.rm = TRUE)
```

Arguments

x	a numeric vector.
na.rm	logical, whether to remove NA values.

Details

This function was borrowed from http://stackoverflow.com/questions/5468280/scale-a-series-between-two-points-in-r/5468527#5468527 and adapted to handle also missing values.

Value

A numeric vector of the same length as the input, now with the values ranging from 0 to 1.

Author(s)

A. Marcia Barbosa

See Also

standard01

Examples

```
range01(0:10)

range01(-12.3 : 21.7)
```

RMSE *Root mean square error*

Description

This function computes the root mean square error of a model object or a set of observed and predicted values or maps. THIS FUNCTION IS NOW DEPRECATED, AS THE RMSE HAS BEEN INCLUDED ALONG OTHER METRICS IN THE NEW `errorMeasures` FUNCTION.

Usage

```
RMSE(model = NULL, obs = NULL, pred = NULL, na.rm = TRUE, rm.dup = FALSE,
verbosity = 2)
```

Arguments

`model`	optional model object of class implemented in `mod2obspred`. If this argument is provided, 'obs' and 'pred' will be extracted with that function. Alternatively, you can input the 'obs' and 'pred' arguments instead of 'model'.
`obs`	alternatively to 'model' and together with 'pred', a numeric vector of observed values of the response variable. Alternatively (and if 'pred' is a 'SpatRaster'), a two-column matrix or data frame containing, respectively, the x (longitude) and y (latitude) coordinates of presence points, in which case the 'obs' vector of presences and absences will be extracted with `ptsrast2obspred`. This argument is ignored if 'model' is provided.
`pred`	alternatively to 'model' and together with 'obs', a vector with the corresponding predicted values, of the same length and in the same order as 'obs'. Alternatively (and if 'obs' is a set of point coordinates), a 'SpatRaster' map of the predicted values for the entire evaluation region, in which case the 'pred' vector will be extracted with `ptsrast2obspred`. This argument is ignored if 'model' is provided.
`na.rm`	Logical value indicating whether missing values should be ignored in computations. Defaults to TRUE.
`rm.dup`	If `TRUE` and if 'pred' is a SpatRaster and if there are repeated points within the same pixel, a maximum of one point per pixel is used to compute the presences. See examples in `ptsrast2obspred`. The default is FALSE.
`verbosity`	integer specifying the amount of messages to display. Defaults to the maximum implemented; lower numbers (down to 0) decrease the number of messages.

Details

The root mean square error is computed as the square root of the mean of the squared differences between observed and predicted values. It is (approximately) the same as the standard deviation of the model residuals (prediction errors), i.e., a measure of how spread out these residuals are, or how concentrated the observations are around the model prediction line. The smaller the RMSE, the better.

Value

The function returns a numeric value indicating the root mean square error of the model predictions.

Author(s)

A. Marcia Barbosa

References

Kenney J.F. & Keeping E.S. (1962) Root Mean Square. "Mathematics of Statistics", 3rd ed. Princeton, NJ: Van Nostrand, pp. 59-60.

See Also

errorMeasures, plotGLM, pseudoRsq, Dsquared

Examples

```
# load sample models:
data(rotif.mods)

# choose a particular model to play with:
mod <- rotif.mods$models[[1]]

RMSE(model = mod)


# you can also use RMSE with vectors of observed and predicted values
# instead of with a model object:

presabs <- mod$y
prediction <- mod$fitted.values

RMSE(obs = presabs, pred = prediction)


# 'obs' can also be a table of presence point coordinates
# and 'pred' a SpatRaster of predicted values
```

---

rotif.mods *Rotifer distribution models*

Description

A set of generalized linear models of rotifer species distributions on TDWG level 4 regions of the world (Fontaneto et al. 2012), together with their predicted values. Mind that these models are provided just as sample data and have limited application, due to limitations in the underlying distribution records. See Details for more information.

Usage

```
data(rotif.mods)
```

Format

A list of 2 elements:

$ predictions: a data.frame with 291 observations of 60 variables, namely the presence probability (P) and environmental favourability (F) for each of 30 species of rotifers, obtained from the rotif.env dataset in the 'fuzzySim' R-Forge package

$ models: a list of the 30 generalized linear model (glm) objects which generated those predictions.

Details

These models were obtained with the 'multGLM' function and the `rotif.env` dataset from R-Forge package 'fuzzySim' using the following code:

require(fuzzySim)

data(rotif.env)

rotif.mods <- multGLM(data = rotif.env, sp.cols = 18:47, var.cols = 5:17, step = FALSE, trim = TRUE)

See package 'fuzzySim' (currently available on R-Forge at http://fuzzysim.r-forge.r-project.org) for more information on the source data that were used to build these models.

References

Fontaneto D., Barbosa A.M., Segers H. & Pautasso M. (2012) The 'rotiferologist' effect and other global correlates of species richness in monogonont rotifers. Ecography, 35: 174-182.

Examples

```
data(rotif.mods)
head(rotif.mods$predictions)
rotif.mods$models[[1]]
```

RsqGLM *R-squared measures for GLMs*

Description

This function is DEPRECATED, please use pseudoRsq instead.

Usage

```
RsqGLM(model = NULL, obs = NULL, pred = NULL, use = "pairwise.complete.obs",
plot = TRUE, plot.type = "lollipop", na.rm = TRUE, rm.dup = FALSE,
verbosity = 2, ...)
```

Arguments

model	see pseudoRsq
obs	see pseudoRsq
pred	see pseudoRsq
use	see pseudoRsq
plot	see pseudoRsq
plot.type	see pseudoRsq
na.rm	see pseudoRsq
rm.dup	see pseudoRsq
verbosity	see pseudoRsq
...	see pseudoRsq

Details

See pseudoRsq.

Value

See pseudoRsq.

Author(s)

A. Marcia Barbosa

Examples

```
# This function is DEPRECATED, please see ?pseudoRsq instead.
```

---

similarity *Similarity measures*

Description

This function computes similarity indices for evaluating the classification accuracy of a species distribution (or ecological niche, or bioclimatic envelope...) model against observed presence-absence data, upon the choice of a threshold value above which the model is considered to predict that the species is expected to be present rather than absent. These metrics were proposed for model evaluation by Li & Guo (2013) and Leroy et al. (2018) – see Details.

Usage

```
similarity(model = NULL, obs = NULL, pred = NULL, thresh,
measures = modEvAmethods("similarity"), simplif = FALSE, pbg = FALSE,
plot = TRUE, plot.type = "lollipop", plot.ordered = FALSE, verbosity = 2,
interval = 0.01, quant = 0, na.rm = TRUE, rm.dup = FALSE, ...)
```

Arguments

model	optional binary-response model object of class "glm", "gam", "gbm", "randomForest" or "bart". If this argument is provided, 'obs' and 'pred' will be extracted with mod2obspred. Alternatively, you can input the 'obs' and 'pred' arguments instead of 'model'.
obs	alternatively to 'model' and together with 'pred', a numeric vector of observed presences (1) and absences (0) of a binary response variable. Alternatively (and if 'pred' is a 'SpatRaster'), a two-column matrix or data frame containing, respectively, the x (longitude) and y (latitude) coordinates of the presence points, in which case the 'obs' vector will be extracted with ptsrast2obspred. This argument is ignored if 'model' is provided.
pred	alternatively to 'model' and together with 'obs', a vector with the corresponding predicted values of presence probability, habitat suitability, environmental favourability or alike. Must be of the same length and in the same order as 'obs'. Alternatively (and if 'obs' is a set of point coordinates), a 'SpatRaster' map of the predicted values for the entire evaluation region, in which case the 'pred' vector will be extracted with ptsrast2obspred. This argument is ignored if 'model' is provided.
thresh	threshold to separate predicted presences from predicted absences in 'model' or 'pred'; can be a numeric value between 0 and 1, or any of the options provided with `modEvAmethods("getThreshold")`. See Details in getThreshold for a description of the available options, and also Details below for a more informed choice.
measures	character vector of the similarity indices to use. By default, all metrics available through `modEvAmethods("similarity")` are included.
simplif	logical, whether to calculate a faster, simplified version. Used internally by other functions in the package. Defaults to FALSE.
pbg	logical value to pass to inputMunch indicating whether to use presence/background (rather than presence/absence) data. Default FALSE.
plot	logical, whether to produce a bar chart or (by default) a lollipop chart of the calculated measures. Defaults to TRUE.
plot.type	character value indicating the type of plot to produce (if plot=TRUE). Can be "lollipop" (the default) or "barplot".
plot.ordered	logical, whether to plot the measures in decreasing order rather than in input order. Defaults to FALSE.
verbosity	integer specifying the amount of messages to display. Defaults to the maximum implemented; lower numbers (down to 0) decrease the number of messages.
interval	Numeric value, used if 'thresh' is a threshold optimization method such as "maxKappa" or "maxTSS", indicating the interval between the thresholds to test. The default is 0.01. Smaller values may provide more precise results but take longer to compute.
quant	Numeric value indicating the proportion of presences to discard if thresh="MTP" (minimum training presence). With the default value 0, MTP will be the threshold at which all observed presences are classified as such; with e.g. quant=0.05, MTP will be the threshold at which 5% presences will be classified as absences.

na.rm	Logical value indicating whether missing values should be ignored in computations. Defaults to TRUE.
rm.dup	If TRUE and if 'pred' is a SpatRaster and if there are repeated points within the same pixel, a maximum of one point per pixel is used to compute the presences. See examples in ptsrast2obspred. The default is FALSE.
...	additional arguments to be passed to the plot function, e.g. ylim=c(0, 1).

Details

Commonly used threshold-based metrics of model evaluation, such as the True Skill Statistic (TSS) implemented in the threshMeasures function, are conditioned by species prevalence in the modelled sample. To overcome this, Leroy et al. (2018) propose using the similary indices of Sorensen and Jaccard for model evaluation, which they show to be (unlike the TSS) independent of prevalence. This function implements such indices in a model evaluation context.

Leroy et al. (2018) point out that Sorensen's index is equivalent to the F-measure (or F1 score, which is also implemented in the threshMeasures function), and that Jaccard's index is half the proxy of the F-measure previously proposed by Li & Guo (2013) for evaluating presence-background models.

Value

If 'simplif=TRUE', the output is a numeric matrix with the name and value of each measure. If 'simplif=FALSE' (the default), the ouptut is a list with the following components:

N	the number of observations (records) in the analysis.
Threshold	the threshold value used to calculate the 'measures'.
similarity	a numeric matrix with the name and value of each measure.

Author(s)

A. Marcia Barbosa

References

Leroy B., Delsol R., Hugueny B., Meynard C.M., Barhoumi C., Barbet-Massin M. & Bellard C. (2018) Without quality presence-absence data, discrimination metrics such as TSS can be misleading measures of model performance. Journal of Biogeography 45(9):1994-2002

Li W. & Guo Q. (2013) How to assess the prediction accuracy of species presence-absence models without absence data? Ecography 36(7):788-799

See Also

threshMeasures, optiThresh

Examples

```
# load sample models:
data(rotif.mods)

# choose a particular model to play with:
mod <- rotif.mods$models[[1]]

similarity(model = mod, thresh = 0.5)

similarity(model = mod, thresh = 0.5, simplif = TRUE, ylim = c(0, 1))

similarity(model = mod, thresh = "maxJaccard")
# or thresh = "maxTSS", "MTP", etc.

# you can also use similarity with vectors of observed and
# predicted values instead of with a model object:

similarity(obs = mod$y, pred = mod$fitted.values, thresh = "maxJaccard")


# 'obs' can also be a table of presence point coordinates
# and 'pred' a SpatRaster of predicted values
```

---

standard01 *Standardize to 0-1 (or vice-versa)*

Description

This function converts the score of a measure that ranges from -1 to 1 (e.g. a kappa or TSS value obtained for a model) into its (linearly) corresponding value in 0-to-1 scale, so that it can be compared directly with measures that range between 0 and 1 (such as CCR or AUC). It can also perform the conversion in the opposite direction.

Usage

```
standard01(score, direction = c("-1+1to01", "01to-1+1"))
```

Arguments

score	numeric value indicating the score of the measure of interest.
direction	character value indicating the direction in which to perform the standardization. The default, "-1+1to01", can be switched to "01to-1+1".

Details

While most of the threshold-based measures of model evaluation range theoretically from 0 to 1, some of them (such as Cohen's kappa and the true skill statistic, TSS) may range from -1 to 1 (Allouche et al. 2006). Thus, the values of different measures may not be directly comparable (Barbosa 2015). We do not usually get negative values of TSS or kappa (nor values under 0.5 for CCR

or AUC, for example) because that only happens when model predictions perform worse than random guesses; still, such values are mathematically possible, and can occur e.g. when extrapolating models to regions where where the species-environment relationships differ. This standardization is included as an option in the threshMeasures function.

Value

The numeric value of 'score' when re-scaled to the 0-to-1 (or to the -1 to +1) scale.

Note

Note that this is not the same as re-scaling a vector so that it ranges between 0 and 1, which is done by range01.

Author(s)

A. Marcia Barbosa

References

Allouche O., Tsoar A. & Kadmon R. (2006) Assessing the accuracy of species distribution models: prevalence, kappa and the true skill statistic (TSS). Journal of Applied Ecology 43: 1223-1232

Barbosa, A.M. (2015) Re-scaling of model evaluation measures to allow direct comparison of their values. The Journal of Brief Ideas, 18 Feb 2015, DOI: 10.5281/zenodo.15487

See Also

threshMeasures, range01

Examples

```
standard01(0.6)

standard01(0.6, direction = "-1+1to01")

standard01(0.6, direction = "01to-1+1")
```

---

threshMeasures *Threshold-based measures of model evaluation*

Description

This function calculates a number of measures for evaluating the classification accuracy of a species distribution (or ecological niche, or bioclimatic envelope) model against observed presence-(pseudo)absence data, or against presence-background data, upon the choice of a threshold value above which the model is considered to predict the species to be present.

Usage

```
threshMeasures(model = NULL, obs = NULL, pred = NULL, thresh,
measures = modEvAmethods("threshMeasures")[-grep("OddsRatio",
modEvAmethods("threshMeasures"))], simplif = FALSE, pbg = FALSE, plot = TRUE,
plot.type = "lollipop", ylim = "auto0", plot.ordered = FALSE, standardize = TRUE,
verbosity = 2, interval = 0.01, quant = 0, na.rm = TRUE, rm.dup = FALSE, ...)
```

Arguments

model	optional binary-response model object of class "glm", "gam", "gbm", "randomForest" or "bart". If this argument is provided, 'obs' and 'pred' will be extracted with mod2obspred. Alternatively, you can input the 'obs' and 'pred' arguments instead of 'model'.
obs	alternatively to 'model' and together with 'pred', a numeric vector of observed presences (1) and absences (0) of a binary response variable. Alternatively (and if 'pred' is a 'SpatRaster'), a two-column matrix or data frame containing, respectively, the x (longitude) and y (latitude) coordinates of the presence points, in which case the 'obs' vector will be extracted with ptsrast2obspred. This argument is ignored if 'model' is provided.
pred	alternatively to 'model' and together with 'obs', a vector with the corresponding predicted values of presence probability, habitat suitability, environmental favourability or alike. Must be of the same length and in the same order as 'obs'. Alternatively (and if 'obs' is a set of point coordinates), a 'SpatRaster' map of the predicted values for the entire evaluation region, in which case the 'pred' vector will be extracted with ptsrast2obspred. This argument is ignored if 'model' is provided.
thresh	threshold to separate predicted presences from predicted absences in 'model' or 'pred'; can be a numeric value between 0 and 1, or any of the options provided with modEvAmethods("getThreshold"). See Details in getThreshold for a description of the available options, and also Details below for a more informed choice.
measures	character vector of the evaluation metrics to use. By default, all metrics available through modEvAmethods("threshMeasures") are included, except for "OddsRatio" which usually yields overly large values that stand out in the plot.
simplif	logical value (default FALSE) indicating whether to calculate a faster, simplified version. Used internally by other functions in the package.
pbg	logical value to pass to inputMunch indicating whether to use presence/background (rather than presence/absence) data. Default FALSE.
plot	logical, whether to produce a bar chart or (by default) a lollipop chart of the calculated measures. Defaults to TRUE.
plot.type	character value indicating the type of plot to produce (if plot=TRUE). Can be "lollipop" (the default) or "barplot".
ylim	either a numeric vector of length 2 specifying the limits (minimum, maximum) for the y axis; or "auto" to fit the y axis to the existing minimum and maximum

values; or "auto0" (the new default) to fit the top of the y axis to the maximum existing values, and the bottom to zero or the minimum existing value (whichever is lower).

`plot.ordered` logical, whether to plot the measures in decreasing order rather than in input order. Defaults to FALSE.

`standardize` logical, whether to change measures that may range between -1 and +1 (namely kappa, TSS, ORSS and SEDI) to their corresponding value in the 0-to-1 scale (skappa and sTSS), so that they can compare directly to other measures (see `standard01`). The default is TRUE, but a message is displayed to inform the user about it.

`verbosity` integer specifying the amount of messages to display. Defaults to the maximum implemented; lower numbers (down to 0) decrease the number of messages.

`interval` Numeric value, used if 'thresh' is a threshold optimization method such as "maxKappa" or "maxTSS", indicating the interval between the thresholds to test. The default is 0.01. Smaller values may provide more precise results but take longer to compute.

`quant` Numeric value indicating the proportion of presences to discard if thresh="MTP" (minimum training presence). With the default value 0, MTP will be the threshold at which all observed presences are classified as such; with e.g. quant=0.05, MTP will be the threshold at which 5% presences will be classified as absences.

`na.rm` Logical value indicating whether missing values should be ignored in computations. Defaults to TRUE.

`rm.dup` If `TRUE` and if 'pred' is a SpatRaster and if there are repeated points within the same pixel, a maximum of one point per pixel is used to compute the presences. See examples in `ptsrast2obspred`. The default is FALSE.

`...` additional arguments to be passed to the `plot` function.

Details

The metrics implemented in this function are based on the confusion (or contingency) matrix, and they are described in dedicated publications (e.g. Fielding & Bell 1997; Liu et al. 2011; Barbosa et al. 2013; Wunderlich et al. 2019). All of them require a threshold value to separate continuous into binary predictions.

The threshold value can be chosen according to a number of criteria (see e.g. Liu et al. 2005, 2013; Jimenez-Valverde & Lobo 2007; Nenzen & Araujo 2011). You can choose a fixed numeric value, or set 'thresh' to "preval" (species' prevalence or proportion of presences **in the data input to this function**), or calculate optimal threshold values according to different criteria with the `getThreshold` function (see also `optiThresh` and `optiPair`). If you are using "environmental favourability" as input 'pred' data (Real et al. 2006; see 'Fav' function in R package **fuzzySim**), then the 0.5 threshold equates to using training prevalence in presence probability as given by logistic regression (GLM with binomial error distribution and logit link function).

While most of these threshold-based measures range from 0 to 1, some of them (such as kappa, TSS, ORSS and SEDI) may range from -1 to 1 (Allouche et al. 2006), so their raw scores are not directly comparable. 'threshMeasures' includes an option (used by default) to standardize these measures to the [0, 1] scale (Barbosa 2015) using the `standard01` function, so that you obtain the standardized versions skappa, sTSS, sORSS and sSEDI.

Note that, in other disciplines such as meteorology and medicine, the TSS is also termed Hanssen and Kuipers discriminant (HKD), Peirce's skill score (PSS), or Youden's J (Youden, 1950; WWRP/WGNE Joint Working Group on Forecast Verification Research, 2017).

This function can also be used to calculate the agreement between different presence-absence (or other types of binary) data, as e.g. Barbosa et al. (2012) did for comparing mammal distribution data from atlas and range maps. Notice, however, that some of these measures, such as TSS or NMI, are not symmetrical (obs vs. pred is different from pred vs. obs).

Note that most of these metrics are correlated with the species' prevalence (proportion of presences) in the modelled sample, so they can over-inflate the perception of model accuracy for restricted species (as does the AUC of the ROC curve). This is the case even for metrics initially alleged to be prevalence-independent, such as TSS (Allouche et al. 2006), as was later shown for real-world data (e.g. Somodi et al. 2017). See similarity for alternative, less prevalence-dependent metrics.

Value

If 'simplif=TRUE', the output is a numeric matrix with the name and value of each measure. If 'simplif=FALSE' (the default), the ouptut is a list with the following components:

`N` the number of observations (records) in the analysis

`Prevalence` the prevalence (proportion of presences) in 'obs'

`Threshold` the threshold value used to calculate the 'measures'

`ConfusionMatrix`
the confusion matrix obtained with the used threshold

`ThreshMeasures` a numeric matrix with the name and value of each of the measures (described in Fielding & Bell 1997 and/or Liu et al. 2011, except where otherwise stated), including:

- CCR: Correct Classification Rate
- Misclass: Misclassification (i.e. incorrect classification) rate
- Sensitivity: proportion of recorded presences correctly predicted as such, i.e. true positive rate
- Specificity: proportion of absences correctly predicted as such, i.e. true negative rate
- Omission: Proportion of missed presences, i.e. proportion of recorded presences incorrectly predicted as absences
- Commission: Proportion of absences incorrectly predicted as presences
- Precision: Proportion of predicted presences actually recorded as presences
- Recall: same as sensitivity (they are both here to match the terminology familiar to different audiences)
- PPP: Positive Predictive Power; same as precision (they are both here to match the terminology familiar to different audiences)
- NPP: Negative Predictive Power, i.e. proportion of predicted absences actually recorded as absences
- UPR: Under-Prediction Rate (Barbosa et al., 2013)
- OPR: Over-Prediction Rate (Barbosa et al., 2013)
- PPI: Potential Presence Increment (Barbosa et al., 2013)

- PAI: Potential Absence Increment (Barbosa et al., 2013)
- kappa: Cohen's kappa, optionally standardized to 'skappa' to fit the [0, 1] scale
- TSS: True Skill Statistic (Allouche et al. 2006), optionally standardized to 'sTSS' to fit the [0, 1] scale
- NMI: Normalized Mutual Information (Forbes, 1995)
- F1score: harmonic mean of precision and recall
- ORSS: Odds Ratio Skill Score (Wunderlich et al., 2019), optionally standardized to 'sORSS' to fit the [0, 1] scale
- SEDI: Symmetric Extremal Dependence Index (Wunderlich et al., 2019), optionally standardized to 'sSEDI' to fit the [0, 1] scale

Note

Some of the implemented measures (e.g. NMI, UPR, OPR, PPP, NPP) cannot be calculated for thresholds at which there are zeros in (some parts of) the confusion matrix, so they can produce NaN values.

Author(s)

A. Marcia Barbosa

References

Allouche O., Tsoar A. & Kadmon R. (2006) Assessing the accuracy of species distribution models: prevalence, kappa and the true skill statistic (TSS). Journal of Applied Ecology 43: 1223-1232.

Barbosa, A.M. (2015) Re-scaling of model evaluation measures to allow direct comparison of their values. The Journal of Brief Ideas, 18 Feb 2015, DOI: 10.5281/zenodo.15487

Barbosa A.M., Estrada A., Marquez A.L., Purvis A. & Orme C.D.L. (2012) Atlas versus range maps: robustness of chorological relationships to distribution data types in European mammals. Journal of Biogeography 39: 1391-1400

Barbosa A.M., Real R., Munoz A.R. & Brown J.A. (2013) New measures for assessing model equilibrium and prediction mismatch in species distribution models. Diversity and Distributions 19: 1333-1338

Fielding A.H. & Bell J.F. (1997) A review of methods for the assessment of prediction errors in conservation presence/absence models. Environmental Conservation 24: 38-49

Forbes, A.D. (1995) Classification-algorithm evaluation: five performance measures based on confusion matrices. Journal of Clinical Monitoring, 11: 189-206

Guisan, A., Thuiller, W. & Zimmermann, N.E. (2017) Measuring Model Accuracy: Which Metrics to Use?, in: Habitat Suitability and Distribution Models: With Applications in R, Ecology, Biodiversity and Conservation. Cambridge University Press, Cambridge, pp. 241-269. https://doi.org/10.1017/9781139028271.022

Hirzel, A.H. & Arlettaz, R. (2003) Modeling habitat suitability for complex species distributions by environmental-distance geometric mean. Environ Manage 32, 614-623. https://doi.org/10.1007/s00267-003-0040-3

Jimenez-Valverde A. & Lobo J.M. (2007) Threshold criteria for conversion of probability of species presence to either-or presence-absence. Acta Oecologica 31: 361-369

Liu C., Berry P.M., Dawson T.P. & Pearson R.G. (2005) Selecting thresholds of occurrence in the prediction of species distributions. Ecography 28: 385-393

Liu C., White M. & Newell G. (2011) Measuring and comparing the accuracy of species distribution models with presence-absence data. Ecography 34: 232-243

Liu C., White M. & Newell G. (2013) Selecting thresholds for the prediction of species occurrence with presence-only data. Journal of Biogeography, 40: 778-789

Nenzen H.K. & Araujo M.B. (2011) Choice of threshold alters projections of species range shifts under climate change. Ecological Modelling 222: 3346-3354

Real R., Barbosa A.M. & Vargas J.M. (2006) Obtaining environmental favourability functions from logistic regression. Environmental and Ecological Statistics 13: 237-245

Somodi I., Lepesi N., Botta-Dukat Z. (2017) Prevalence dependence in model goodness measures with special emphasis on true skill statistics. Ecology and Evolution, 7(3):863-872. doi: 10.1002/ece3.2654

WWRP/WGNE Joint Working Group on Forecast Verification Research (2017). Forecast Verification methods Across Time and Space Scales. 7th International Verification Methods Workshop, https://www.cawcr.gov.au/projects/verification/

Wunderlich R.F., Lin Y.-P., Anthony J., Petway J.R. (2019) Two alternative evaluation metrics to replace the true skill statistic in the assessment of species distribution models. Nature Conservation 35: 97-116

Youden, W.J. (1950) Index for rating diagnostic tests. Cancer. 3: 32-35

See Also

similarity, optiThresh, optiPair, AUC

Examples

```
# load sample models:
data(rotif.mods)

# choose a particular model to play with:
mod <- rotif.mods$models[[1]]

threshMeasures(model = mod, simplif = TRUE, thresh = 0.5)

threshMeasures(model = mod, thresh = "preval")

threshMeasures(model = mod, plot.ordered = TRUE, thresh = "preval")

threshMeasures(model = mod, measures = c("CCR", "TSS", "kappa"),
thresh = "preval")

threshMeasures(model = mod, plot.ordered = TRUE, thresh = "preval")


# you can also use threshMeasures with vectors of observed and
# predicted values instead of with a model object:
```

```
threshMeasures(obs = mod$y, pred = mod$fitted.values, thresh = "preval")


# 'obs' can also be a table of presence point coordinates
# and 'pred' a SpatRaster of predicted values
```

`varImp` *Variable importance.*

Description

This function computes, and optionally plots, variable importance for an input model object of an implemented class. Note that "importance" is a vague concept which can be measured in different ways, and not all importance type options are available for all model classes (see Arguments and Details).

Usage

```
varImp(model, imp.type = "each", relative = TRUE, reorder = TRUE,
group.cats = FALSE, n.per = 10, data = NULL, n.trees = 100, plot = TRUE,
plot.type = "lollipop", error.bars = "sd", ylim = "auto0",
col = c("steelblue4", "coral2"), plot.points = TRUE, legend = TRUE,
grid = TRUE, verbosity = 2, ...)
```

Arguments

`model`	a (binary-response) model object of class "glm" (of package **stats**), "Gam" (of package **gam**), "gam" (of package **mgcv**), "gbm" (of package **gbm**), "GBMFit" (of package **gbm3**), "randomForest" (of package **randomForest**), "maxnet" (of package **maxnet**), "bart" (of package **dbarts**), "pbart" or "lbart" (of package **BART**), or a list produced by function "flexBART" or "probit_flexBART" of package **flexBART**.
`imp.type`	character value indicating the type of variable importance to compute, i.e. the metric with which importance is measured. Partial argument matching is used. Implemented options are: • "each" (the default), to extract the measure provided by each model object or summary (note that this is different across model classes, and not available for al of them – see Details) • "permutation", to randomly shuffle each variable in turn (a given number of times) and average the root mean squared difference between the actual model predictions and those obtained with the shuffled variable. Note this can be considerably slower, especially for computationally intensive models. Use `set.seed()` first if you want exactly reproducible results. This type will fail for model objects that do not store the predictor values, e.g. `maxnet` or `randomForest`.

`relative` logical value (default TRUE) indicating whether to divide the absolute importance values by their total sum, to get a measure of relative variable importance ranging from 0 to 1. Applies when imp.type="permutation", or to GLM and BART models when imp.type="each".

`reorder` logical value indicating whether to sort the variables in decreasing order of importance. The default is TRUE. If set to FALSE, the variables retain their input order.

`group.cats` logical value indicating whether to aggregate all factor levels of each (one-hot encoded) categorical variable into a single variable, by summing up their proportions of branches used. Used if 'model' is of class 'bart', 'pbart' or 'lbart', in whose outputs the contributions of categorical variables are split by their factor levels. The default is FALSE. NOTE that this may incorrectly group variables that have the same name with a different numeric suffix (e.g. "o2" and "o3", or "soil_type_1", "soil_type_2"), so revise your results if you set this to TRUE!

`n.per` (if imp.type="permutation") number of permutations for each variable. The default is 10.

`data` (if imp.type="permutation") matrix or data frame with the predictor variables for which to compute permutation importance. The default is to extract data from the model object if it contains this information (e.g. for models of class "glm", "GBMFit", or "bart" computed with keeptrees=TRUE), or an error message otherwise, in which case the user needs to provide this argument. Note that only the variables actually included in the model should be provided in 'data'.

`n.trees` (if imp.type="permutation") argument required by `predict.GBMFit()` if 'model' is of class "GBMFit". The default is 100.

`plot` logical value indicating whether to produce a plot with the results. The default is TRUE.

`plot.type` (if plot=TRUE) character value indicating the type of plot to produce. Can be "`lollipop`" (the default), "`barplot`", or "`boxplot`". Note that the latter is only useful when 'model' contains several importance values per variable (e.g. for models of class "bart"). Partial argument matching is used.

`error.bars` character value indicating the type of error metric to compute (and plot, if plot=TRUE) if the input contains the necessary information (i.e., for Bayesian models like BART) and if the 'plot.type' is appropriate (i.e. "lollipop" or "barplot"). Can be "sd" (the default) for the standard deviation; "range" for the minimum and maximum value across the ones available; a numeric value between 0 and 1 for the corresponding confidence interval (e.g. 0.95 for 95%), computed with `quantile`; or NA for no error bars.

`ylim` (if plot=TRUE) either a numeric vector of length 2 specifying the limits (minimum, maximum) for the y axis; or "auto" to fit the y axis to the existing minimum and maximum values; or "auto0" (the new default) to fit the top of the y axis to the maximum existing values, and the bottom to zero.

`col` (if plot=TRUE) character or integer vector of length 1 or 2 specifying the plotting colours for the variables with positive and negative effect on the response, when this info is available (e.g. for models of class "glm" when imp.type="each").

`plot.points` (if plot=TRUE and 'mod' includes several possible importance values per variable, i.e. BART models) logical value indicating whether or not to add to the plot

the individual importance points (rather than just the mean importance value, and the error bar if error.bars=TRUE) for each variable. By default it is TRUE, following Weissgerber et al. (2015).

`legend` logical value indicating whether or not to draw a legend. Used only if plot=TRUE and if the output includes negative values (i.e., if 'model' is of class 'glm' and imp.type="each" and there are variables with positive and negative coefficients).

`grid` (if plot=TRUE) logical, whether or not to add a grid to the plot. The default is TRUE.

`verbosity` integer indicating the amount of messages to display. Defaults to the maximum implemented; lower numbers (down to 0) decrease the number of messages.

`...` (if plot=TRUE) additional arguments that can be passed to the plotting function (depending on 'plot.type'), e.g. 'main', 'horiz' (or 'horizontal' for `boxplot`), 'cex.axis' (for `lollipop` or `boxplot`), or 'cex.names' (for `barplot`).

Details

Variable importance is a non-objective characteristic which can be measured in a variety of ways – e.g., the weight of a variable in a model (e.g. how strong its coefficient for in regression, or how many times it is used for decision trees); how much worse the model would be without it (according to a given performance metric); or how different the predictions would be if the variable were shuffled or not used. If you compute variable importance with different methods (e.g. with the functions suggested in the "See also" section), you are likely to get varied results.

In this function, when imp.type="each" (the default), variable importance in a model of class "glm" (obtained with the `glm` function) can be measured by the magnitude of the absolute z-value test statistic, which is provided with `summary(model)`. The 'varImp' function outputs either the absolute z value of each variable or, if relative=TRUE (the default), the relative z value, obtained by dividing the absolute z value by the sum of absolute z values in the model. In the plot (by default), different colours are used for variables with positive and negative relationships with the response.

If the input model is of class "Gam" of the **gam** package or "gam" of the **mgcv** package, variable importance (when imp.type="each") is measured by the Chi-squared values of the smooth terms. Note this is not a very consensual metric, and a warning is produced acknowledging that the values may depend on the effective degrees of freedom (EDF) and scaling of each term.

If the input model is of class "gbm" of the **gbm** package, variable importance is obtained from `summary.gbm(model)` and divided by 100 to get the result as a proportion rather than a percentage (for consistency). See the help file of that function for details.

If the input model is of class "randomForest" of the **randomForest** package, variable importance is obtained with `model$importance`. See the help file of randomForest for details.

If the input model is of class "bart" of the **dbarts** package, or of class "pbart" or "lbart" of the **BART** package, or a list produced by function "probit_flexBART" of the **flexBART** package, variable importance is obtained as the mean (if relative=TRUE, the default) or the total number (if relative=FALSE) of regression tree splits where each variable is used. If 'error.bars' is not NA, the error is also computed according to the specified metric ("sd" or standard deviation by default).

If `imp.type` is set instead to "permutation", each variable is randomly shuffled `n.per` times, and the root mean squared difference is computed between the model predictions and those with the shuffled variable.

Value

This function outputs, and optionally plots, a named numeric vector of variable importance, as measured by 'imp.type' (see Details). If the model is Bayesian (BART) and 'error.bars' is not NA, the output is a row-named data frame with the mean as well as the lower and upper bounds of the error bars (according to the specified error metric - the default is standard deviation) of variable importance.

Author(s)

A. Marcia Barbosa

References

Greenwell B.M. & Boehmke B.C. (2020) Variable Importance Plots - An Introduction to the vip Package". The R Journal, 2020: https://journal.r-project.org/articles/RJ-2020-013/

Weissgerber T.L., Milic N.M., Winham S.J. & Garovic V.D. (2015) Beyond Bar and Line Graphs: Time for a New Data Presentation Paradigm. PLOS Biol 13:e1002128. https://doi.org/10.1371/JOURNAL.PBIO.1002128

See Also

plotCoeffs; summary.glm; caret::varImp; embarcadero::varimp; enmpa::var_importance; predicts::varImportance; biomod2::bm_VariablesImportance; iml::FeatureImp; ingredients::feature_import mmpf::permutationImportance; package **varImp**; package **vip**; package **relaimpo**; numerous packages that implement Shapley values

Examples

```
# load sample models:

data(rotif.mods)


# choose a particular model to play with:

mod <- rotif.mods$models[[2]]


# get variable importance for this model:

varImp(model = mod,
       cex.axis = 0.6)

varImp(model = mod,
      imp.type = "perm",
      cex.axis = 0.6)


# change some more parameters:

par(mar = c(3, 9.5, 2.5, 1))
```

```
varImp(model = mod,
       relative = FALSE,
       col = c("darkgreen", "orange"),
       plot.type = "barplot",
       cex.names = 0.85,
       ylim = c(0, 5),
       main = "Variable importance in \n my model",
       horiz = TRUE)
```

---

varPart *Variation partitioning*

Description

This function performs variation partitioning (Borcard et al. 1992) among two factors (e.g. Ribas et al. 2006) or three factors (e.g. Real et al. 2003) for either linear regression models (LM) or generalized linear models (GLM).

Usage

```
varPart(A, B, C = NA, AB, AC = NA, BC = NA, ABC = NA, model.type = NULL,
A.name = "Factor A", B.name = "Factor B", C.name = "Factor C",
model = NULL, groups = NULL, pred.type = "Y", cor.method = "pearson",
return.models = FALSE, plot = TRUE, plot.digits = 3, cex.names = 1.5,
cex.values = 1.2, main = "", cex.main = 2, plot.unexpl = TRUE, colr = FALSE)
```

Arguments

A	numeric value of the R-squared of the regression of the response variable on the variables related to factor 'A'. NOTE: INSTEAD of this and the next 10 arguments, you can use arguments 'model' and 'groups' below.
B	numeric value of the R-squared of the regression of the response variable on the variables related to factor 'B'
C	(optionally, if there are 3 factors) numeric value of the R-squared of the regression of the response on the variables related to factor 'C'
AB	numeric value of the R-squared of the regression of the response on the variables of factors 'A' and 'B' simultaneously
AC	(if there are 3 factors) numeric value of the R-squared of the regression of the response on the variables of factors 'A' and 'C' simultaneously
BC	(if there are 3 factors) numeric value of the R-squared of the regression of the response on the variables of factors 'B' and 'C' simultaneously
ABC	(if there are 3 factors) numeric value of the R-squared of the regression of the response on the variables of factors 'A', 'B' and 'C' simultaneously
model.type	deprecated argument, kept here for back-compatibility
A.name	character string indicating the name of factor 'A'

B.name	character string indicating the name of factor 'B'
C.name	character string indicating the name of factor 'C' (if there are 3 factors)
model	a model object of class 'glm' (for linear models, instead of 'lm' you can use use 'glm' with family=gaussian). If this argument is provided, all previous arguments are ignored, as they are computed instead from 'model' and 'groups'.
groups	data frame with 2 columns, the 1st one containing the names of the variables, and the 2nd one containing the names of the factors in which they should be grouped (e.g. climatic, human, topographic) for the variation partitioning. This argument is required (and only used) if 'model' is provided.
pred.type	character value specifying the type of predictions among which to calculate the R-squared values, or squared correlations. Can be "Y" (the default) for the 'link' function (in the scale of the predictor variables), "P" for using the 'response' (e.g. in the scale of probability for models of family binomial), or "F" for using favourability (i.e., probability after removing the effect of modelled prevalence, as in the 'Fav' function of package **fuzzySim**); see Details. This argument is only used if 'model' is provided.
cor.method	character value to pass to the 'method' argument of cor specifying the correlation coefficient to use. The default is "pearson". This argument is only used if 'model' is provided.
return.models	logical value indicating whether to include in the output the model obtained for each group of variables. The default is FALSE. This argument is only used if 'model' is provided.
plot	logical, whether to plot the variation partitioning diagram. The default is TRUE.
plot.digits	integer value of the number of digits to which to round the values in the plot. The default is 3.
cex.names	numeric value indicating character expansion factor to define the size of the names of the factors displayed in the plot.
cex.values	numeric value indicating character expansion factor to define the size of the values displayed in the plot.
main	optional character string indicating the main title for the plot. The default is empty.
cex.main	numeric value indicating character expansion factor to define the font size of the plot title (if provided).
plot.unexpl	logical value indicating whether the amount of unexplained variation should be included in the plot. The default is TRUE.
colr	logical value indicating whether or not to colour the circles in the plot. The default is FALSE for back-compatibility.

Details

If you have linear models (i.e. GLMs of family Gaussian), input data for 'varPart' are the coefficients of determination (R-squared values) of the linear regressions of the response variable on all the variables in the model, on the variables related to each particular factor, and (when there are 3 factors) on the variables related to each pair of factors. The outputs are the amounts of variance explained exclusively by each factor, the amounts explained exclusively by the overlapping effects of

each pair of factors, and the amount explained by the overlap of the 3 factors if this is the case (e.g. Real et al. 2003). The amount of variation not explained by the complete model is also provided.

If you have generalized linear models (GLMs) such as logistic regression (see `glm`), you have no true R-squared values; inputs can then be the squared coefficients of correlation between the model predictions given by each factor (or pair of factors) and the predictions of the complete model. Predictions can be probability (e.g. Munoz & Real 2006), favourability (Baez et al. 2012, Estrada et al. 2016), or the 'logit' linear predictor (Real et al. 2013); the correltion coefficient can be e.g. Pearson's (Munoz & Real 2006) or Spearman's (Baez et al. 2012). An adjusted R-squared can also be used (De Araujo et al. 2014). In GLMs, the "total variation" (AB or ABC, depending on whether you have two or three factors) is 1 (correlation of the predictions of the complete model with themselves), and output values are not the total amounts of variance (of the response variable) explained by variable groups and their overlaps, but rather their proportional contribution to the total variation explained by the model.

Value

This function returns a data frame indicating the proportion of variance accounted for by each of the factors or groups, and (if 'plot = TRUE') a Venn diagram of the contributions of each factor or overlap. If 'return.models=TRUE', the output includes also the model obtained for each group of variables.

Note

These results derive from arithmetic operations between your input values, and they always sum up to 1; if your input is incorrect, the results will be incorrect as well, even if they sum up to 1.

This function had a bug up to modEvA version 0.8: a badly placed line break prevented the ABC overlap from being calculated correctly. Thanks to Jurica Levatic for pointing this out and helping to solve it!

Oswald van Ginkel also suggested a fix to some plotting awkwardness when using only two factors, and a nice option for colouring the plot. Many thanks!

Author(s)

A. Marcia Barbosa

References

Baez J.C., Estrada A., Torreblanca D. & Real R (2012) Predicting the distribution of cryptic species: the case of the spur-thighed tortoise in Andalusia (southern Iberian Peninsula). Biodiversity and Conservation 21: 65-78

Borcard D., Legendre P., Drapeau P. (1992) Partialling out the spatial component of ecological variation. Ecology 73: 1045-1055

De Araujo C.B., Marcondes-Machado L.O. & Costa G.C. (2014) The importance of biotic interactions in species distribution models: a test of the Eltonian noise hypothesis using parrots. Journal of Biogeography 41: 513-523

Estrada A., Delgado M.P., Arroyo B., Traba J., Morales M.B. (2016) Forecasting Large-Scale Habitat Suitability of European Bustards under Climate Change: The Role of Environmental and Geographic Variables. PLoS ONE 11(3): e0149810

Munoz A.-R. & Real R. (2006) Assessing the potential range expansion of the exotic monk parakeet in Spain. Diversity and Distributions 12: 656-665

Real R., Barbosa A.M., Porras D., Kin M.S., Marquez A.L., Guerrero J.C., Palomo L.J., Justo E.R. & Vargas J.M. (2003) Relative importance of environment, human activity and spatial situation in determining the distribution of terrestrial mammal diversity in Argentina. Journal of Biogeography 30: 939-947

Real R., Romero D., Olivero J., Estrada A. & Marquez A.L. (2013) Estimating how inflated or obscured effects of climate affect forecasted species distribution. PLoS ONE 8: e53646

Ribas A., Barbosa A.M., Casanova J.C., Real R., Feliu C. & Vargas J.M. (2006) Geographical patterns of the species richness of helminth parasites of moles (Talpa spp.) in Spain: separating the effect of sampling effort from those of other conditioning factors. Vie et Milieu 56: 1-8

Examples

```
# if you have a linear model (LM), use (non-adjusted) R-squared values
# for each factor and for their combinations as inputs:

# with 2 factors:

varPart(A = 0.456, B = 0.315, AB = 0.852, A.name = "Spatial",
B.name = "Environmental", main = "Small whale")

varPart(A = 0.456, B = 0.315, AB = 0.852, A.name = "Spatial",
B.name = "Environmental", main = "Small whale", colr = TRUE)


# with 3 factors:

varPart(A = 0.456, B = 0.315, C = 0.281, AB = 0.051, BC = 0.444,
AC = 0.569, ABC = 0.624, A.name = "Spatial", B.name = "Human",
C.name = "Environmental", main = "Small whale")

varPart(A = 0.456, B = 0.315, C = 0.281, AB = 0.051, BC = 0.444,
AC = 0.569, ABC = 0.624, A.name = "Spatial", B.name = "Human",
C.name = "Environmental", main = "Small whale", colr = TRUE)


# if you have a generalized linear model (GLM),
# you can use squared Pearson correlation coefficients of the
# predictions of each factor with those of the complete model:

varPart(A = (-0.005)^2, B = 0.698^2, C = 0.922^2, AB = 0.696^2,
BC = 0.994^2, AC = 0.953^2, ABC = 1, A.name = "Topographic",
B.name = "Climatic", C.name = "Geographic", main = "Big bird")

# but "Unexplained variation" can be deceiving in these cases
# (see Details); try also adding 'plot.unexpl = FALSE'


# if you have a model object and a table classifying the variables into groups:
```

```
data(rotif.mods)
mod <- rotif.mods$models[[2]]

head(mod$model)
vars <- colnames(mod$model)[-1]
vars
var_groups <- data.frame(vars = vars, groups = c("Spatial", "Spatial",
"Climate", "Climate", "Climate", "Human"))
var_groups

varPart(model = mod, groups = var_groups)
varPart(model = mod, groups = var_groups, pred.type = "P", colr = TRUE)
```

Index

∗ **datasets**
rotif.mods, 84
∗ **package**
modEvA-package, 3

applyThreshold, 4
arrangePlots, 7, *61*
arrows, *43*
AUC, 8, *78*, *93*, *95*
axis, *43*

barplot, *44*, *77*, *87*, *91*, *97, 98*
boxplot, *97, 98*
Boyce, 13, *29*, *38*, *42*, *49*, *54*, *68, 69*

confint, *64*
confusionLabel, 16, *20*
confusionMatrix, *18*, 18
cor, *14*, *77*, *101*
cor.test, *68, 69*

density, *70, 71*
Dsquared, 21, *42*, *49*, *65*, *78*

errorMeasures, 23, *83*
evaluate, 25
evenness, 27, *76*

family, *21*

getBins, 28, *36–38*, *53, 54*, *68, 69*
getModEqn, 31
getThreshold, *5, 6*, 32, *53*, *59, 60*, *62*, *73*, *81*, *87*, *91, 92*
ggplot2::geom_pointrange(), *44*
glm, *65*, *85*, *98*, *102*

hist, *70, 71*
HLfit, *16*, *29, 30*, 35, *42*, *49*, *54, 55*, *78*

image, *20*
inputMunch, *5*, *9*, *14*, *22*, *33*, 39, *58*, *61*, *71*, *73*, *87*, *91*

jitter, *74*

ks.test, *71*

layout, *7*
lines, *68*
logLik, *42*
logLike, 40
lollipop, 42, *64*, *77*, *87*, *91*, *97, 98*

maxnet, *96*
MESS, 44, *56, 57*
Miller, *16*, *38*, *42*, 47, *50, 51*, *54*, *78*
MillerCalib, 50
mod2obspred, *5*, *8*, *13*, *17*, *19*, *21*, *23*, *28*, *33*, *35*, *39, 40*, *47*, 52, *58*, *60*, *65*, *67*, *70*, *73*, *75*, *83*, *87*, *91*
modEvA *(*modEvA-package*)*, 3
modEvA-package, 3
modEvAmethods, 53
mtext, *43*
multModEv, *53*, 54

OA, *46*, 56
optiPair, *34*, 57, *62*, *92*, *95*
optiThresh, *5*, *33, 34*, *53*, *59*, 60, *73*, *88*, *92*, *95*

par, *43*, *61*, *73*
plot, *7*, *10*, *14*, *20*, *37*, *43*, *48*, *59*, *61, 62*, *66*, *88*, *92*
plotCoeffs, *43, 44*, 63, *99*
plotGLM, *22*, *24*, 65, *74*, *84*
points, *43*
poMeasures, 67
predDensity, *66*, 70, *74*
predPlot, *66*, *71*, 72
prevalence, *27*, *52*, *73*, 75

pseudoRsq, *22*, *24*, *42*, *49*, *65*, 76, *84–86*
ptsrast2obspred, *5*, *8*, *10*, *13, 14*, *17*, *19*, *21, 22*, *24*, *28, 29*, *33*, *35, 36*, *39–41*, *47, 48*, *58*, *61*, *65, 66*, *68*, *70, 71*, *73*, *77*, *79*, *83*, *87, 88*, *91, 92*

quantile, *29, 30*, *36*, *81*, *97*
quantReclass, 80

randomForest, *96*
range01, 82, *90*
RMSE, *24*, *42*, 83
rotif.mods, 84
round, *10*, *14*, *66*, *68*, *101*
RsqGLM, 85

set.seed, *74*
similarity, *53*, 86, *93*, *95*
standard01, *54*, *82*, 89, *92*
summary, *64*
summary.glm, *99*

threshMeasures, *6*, *12*, *17–20*, *26*, *33, 34*, *53–55*, *59, 60*, *62*, *78*, *88*, *90*, 90

varImp, *52*, *64*, 96
varPart, 100

wilcox.test, *69*, *71*