# Package ‘fuzzySim’

November 23, 2015

**Type** Package

**Title** Fuzzy similarity in species distributions

**Version** 1.6.2

**Date** 2015-11-23

**Author** Barbosa A.M.

**Maintainer** A. Marcia Barbosa <barbosa@uevora.pt>

**Description** Functions to calculate fuzzy versions of species' occurrence patterns based on presence-absence data (including inverse distance interpolation, trend surface analysis and prevalence-independent favourability GLM), and pair-wise fuzzy similarity (based on fuzzy versions of commonly used similarity indices) among those occurrence patterns. Includes also functions for model comparison (overlap, loss and gain of ), and for data preparation, such as obtaining unique abbreviations of species names, converting species lists (long format) to presence-absence tables (wide format), transposing part of a data frame, assessing the false discovery rate, or analysing and dealing with multicollinearity among variables. Includes also sample datasets for providing practical examples.

**License** GPL-3

## R topics documented:

---

`fuzzySim-package` *Fuzzy similarity in species distributions*

---

## Description

Functions to calculate fuzzy versions of species' occurrence patterns based on presence-absence data (including inverse distance interpolation, trend surface analysis and prevalence-independent favourability GLM), and pair-wise fuzzy similarity (based on fuzzy versions of commonly used similarity indices) among those occurrence patterns. Includes also functions for data preparation, such as obtaining unique abbreviations of species names, converting species lists (long format) to presence-absence tables (wide format), transposing part of a data frame, assessing the false discovery rate, or analysing and dealing with multicollinearity among variables. Includes also sample datasets for providing practical examples. A step-by-step illustrated tutorial is available from the package homepage (http://fuzzysim.r-forge.r-project.org).

## Details

| | |
|---|---|
| Package: | fuzzySim |
| Type: | Package |
| Version: | 1.6.2 |
| Date: | 2015-11-23 |
| License: | GPL-3 |

## Author(s)

A. Marcia Barbosa

Maintainer: A. Marcia Barbosa <barbosa@uevora.pt>

## References

Barbosa A.M. (2015) fuzzySim: applying fuzzy logic to binary similarity indices in ecology. Methods in Ecology and Evolution, 6: 853-858.

## See Also

**RMACOQUI**

## Examples

```
data(rotifers)

head(rotifers)


# add column with species name abbreviations:

rotifers$spcode <- spCodes(rotifers$species, sep.species = "_", nchar.gen = 1,
nchar.sp = 5, nchar.ssp = 0)

head(rotifers)


# convert species list (long format) to presence-absence table (wide format):

rotifers.presabs <- splist2presabs(rotifers, sites.col = "TDWG4",
sp.col = "spcode", keep.n = FALSE)

head(rotifers.presabs)


# get 3rd-degree spatial trend surface for each species' distribution:

data(rotif.env)

names(rotif.env)

rotifers.tsa <- multTSA(rotif.env, sp.cols = 18:47,
coord.cols = c("Longitude", "Latitude"), id.col = 1)

head(rotifers.tsa)


# get inverse squared distance to presence for each species:

rotifers.isqd <- distPres(rotif.env, sp.cols = 18:47,
coord.cols = c("Longitude", "Latitude"), id.col = 1, p = 2, inv = TRUE)

head(rotifers.isqd)
```

```
# get prevalence-independent environmental favourability models for each species:

data(rotif.env)

names(rotif.env)

rotifers.fav <- multGLM(data = rotif.env, sp.cols = 18:47, var.cols = 5:17,
id.col = 1, step = FALSE, trim = TRUE, Favourability = TRUE)


# get matrix of fuzzy similarity between rotifer species distributions:

# either based on inverse squared distance to presence:
rot.fuz.sim.mat <- simMat(rotifers.isqd[ , -1], method = "Jaccard")

# or on environmental favourability for presence:
rot.fuz.sim.mat <- simMat(rotifers.fav$predictions[ , 32:61], method = "Jaccard")


head(rot.fuz.sim.mat)


# transpose fuzzy rotifer distribution data to compare
# regional species composition rather than species' distributions:

names(rotifers.isqd)

rot.fuz.reg <- transpose(rotifers.fav$predictions, sp.cols = 32:61,
reg.names = 1)

head(rot.fuz.reg)


# get matrix of fuzzy similarity between (some) regions' species compositions:

reg.fuz.sim.mat <- simMat(rot.fuz.reg[ , 1:100], method = "Jaccard")

head(reg.fuz.sim.mat)
```

---

`corSelect` *Select among correlated variables based on their relationship with the response*

---

## Description

This function calculates pairwise correlations among the variables in a dataset and, among each pair of variables correlated above a given threshold, selects the one with the most significant or most informative bivariate (individual) relationship with the response variable.

## Usage

```
corSelect(data, sp.cols, var.cols, cor.thresh = 0.8, select = "p.value", ...)
```

## Arguments

| | |
|---|---|
| `data` | a data frame containing the response and predictor variables. |
| `sp.cols` | index number of the column of `data` that contains the response (e.g. species) variable. |
| `var.cols` | index numbers of the columns of `data` that contain the predictor variables. |
| `cor.thresh` | threshold value of correlation coefficient above which predictor variables should be excluded. The default is 0.8. |
| `select` | character value indicating the criterion for selecting variables (among those that are correlated) based on their relationship with the response variable. Can be "p.value" (the default) or "AIC". A message is displayed saying whether the result would have been different with the alternative criterion. |
| `...` | additional arguments to pass to `cor`, namely the `method` to use (either `"pearson"`, `"kendall"` or `"spearman"` correlation coefficient; the first is the default) and the way to deal with missing values (`use = "everything"`, `"all.obs"`, `"complete.obs"`, `"na.or.complete"`, or `"pairwise.complete.obs"`). |

## Details

Correlations among variables are problematic in multivariate models, as they inflate the variance of coefficients and thus may bias the interpretation of the effects of those variables on the response (Legendre & Legendre 2012). One of the strategies to circumvent this problem is to eliminate a priori one from each pair of correlated variables, but it is not always straightforward to choose the right variable. This function selects such variables based on their relationship with the response variable, by building a bivariate model of each individual variable against the response and choosing, among each of two correlated variables, the one with the smaller (best) AIC or p-value.

The function uses only the rows of the dataset where the response variable contains finite values against which the predictor variables can be modelled. Rows with NA or NaN in the response variable are excluded from the calculation of correlations among predictor variables).

## Value

This function returns a list of 7 elements:

`high.correlations`
: data frame showing the pairs of input variables that are correlated above the threshold.

`bivariate.significance`
: data frame with the individual AIC (Akaike's Information Criterion) and significance (p-value) of each of the highly correlated variables against the response variable.

`excluded.vars`
: character vector containing the names of the variables to be excluded (i.e., from each highly correlated pair, the variable with larger AIC or p-value against the response).

selected.vars
: character vector containing the names of the variables to be selected (i.e., the non-correlated variables and, from each correlated pair, the variable with smaller AIC or p-value against the response).

selected.var.cols
: integer vector containing the column indices of the variables to be selected in data.

strongest.remaining.corr
: numerical value indicating the highest (absolute) correlation coefficient among the selected variables.

remaining.multicollinearity
: data frame showing the multicollinearity among the selected variables.

## Author(s)

A. Marcia Barbosa

## References

Legendre P. & Legendre L. (2012) Numerical ecology (3rd edition). Elsevier, Amsterdam: 990 pp.

## See Also

multicol, FDR, cor

## Examples

```
data(rotif.env)

corSelect(rotif.env, sp.cols = 46, var.cols = 5:17)

corSelect(rotif.env, sp.cols = 46, var.cols = 5:17, cor.thresh = 0.7)

corSelect(rotif.env, sp.cols = 46, var.cols = 5:17, method = "spearman")
```

---

# distPres *(Inverse) distance to the nearest presence*

---

## Description

This function takes a matrix or data frame containing species presence (1) and absence (0) data and their spatial coordinates (optionally also a pre-calculated distance matrix between all localities), and calculates the (inverse) distance from each locality to the nearest presence locality for each species.

## Usage

```
distPres(data, sp.cols, coord.cols = NULL, id.col = NULL, dist.mat = NULL,
method = "euclidian", suffix = "_D", p = 1, inv = TRUE)
```

## Arguments

| | |
|---|---|
| `data` | a matrix or data frame containing, at least, two columns with spatial coordinates, and one column per species containing their presence (1) and absence (0) data, with localities in rows. |
| `sp.cols` | names or index numbers of the columns containing the species presences and absences in `data`. It must contain only zeros (0) for absences and ones (1) for presences. |
| `coord.cols` | names or index numbers of the columns containing the spatial coordinates in data (in this order, x and y, or longitude and latitude). |
| `id.col` | optionally, the name or index number of a column (to be included in the output) containing locality identifiers in data. |
| `dist.mat` | optionally, if you do not want distances calculated with any of the methods available in `dist`, you may provide a distance matrix calculated elsewhere for the localities in `data`. |
| `method` | the method with which to calculate distances between localities. Available options are those of `dist`. The default is `"euclidian"`. |
| `suffix` | character indicating the suffix to add to the distance columns in the resulting data frame. The default is `"_D"`. |
| `p` | the power to which distance should be raised. The default is 1; use 2 or higher if you want more conservative distances. |
| `inv` | logical value indicating whether distance should be inverted, so that it varies between 0 and 1 and higher values mean closer to presence. The default is `TRUE`, which is adequate as a fuzzy version of presence-absence (for using e.g. with `fuzSim` and `simMat`). In this case, presences maintain the value 1, and inverse distance to presence is calculated only for absence localities. |

## Value

`distPres` returns a matrix or data frame containing the identifier column (if provided in `id.col`) and one column per species containing the distance (inverse squared by default) from each locality to the nearest presence of that species.

## Author(s)

A. Marcia Barbosa

## See Also

`dist`

## Examples

```
data(rotif.env)

head(rotif.env)

names(rotif.env)
```

```
# calculate plain distance to presence:

rotifers.dist <- distPres(rotif.env, sp.cols = 18:47, coord.cols = c("Longitude",
"Latitude"), id.col = 1, p = 1, inv = FALSE, suffix = "_D")

head(rotifers.dist)


# calculate inverse squared distance to presence:

rotifers.invd2 <- distPres(rotif.env, sp.cols = 18:47, coord.cols = c("Longitude",
"Latitude"), id.col = 1, p = 2, inv = TRUE, suffix = "_iDsq")

head(rotifers.invd2)
```

---

Fav *Favourability*

---

## Description

Environmental (prevalence-independent) favourability for a species' presence

## Usage

```
Fav(model = NULL, obs = NULL, pred = NULL, n1n0 = NULL, sample.preval = NULL,
method = "RBV", true.preval = NULL)
```

## Arguments

| | |
|---|---|
| `model` | a model object of class `"glm"` and binomial family. |
| `obs` | a vector of the 1/0 values of the modelled binary variable. This argument is ignored if `model` is provided. |
| `pred` | a vector of predicted probability values for `obs`, given e.g. by logistic regression. This argument is ignored if `model` is provided. |
| `n1n0` | alternatively to `obs`, an integer vector of length 2 providing the total numbers of modelled ones and zeros, in this order. Ignored if `obs` or `model` is provided. |
| `sample.preval` | alternatively to `obs` or `n1n0`, the prevalence (proportion of positive cases) of the modelled binary variable in the modelled data. Ignored if `model` is provided. |
| `method` | either `"RBV"` for the original Real, Barbosa & Vargas (2006) procedure, or `"AT""` for the modification proposed by Albert & Thuiller (2008) (but see Details). |
| `true.preval` | the true prevalence (as opposed to sample prevalence), necessary if you want to use the AT method. |

## Details

Logistic regression (Generalised Linear Model with binomial error distribution and a logit link) is widely used for modelling species' potential distributions using presence/absence data and a set of categorical or continuous predictor variables. However, this GLM incorporates the prevalence (proportion of presences) of the species in the training sample, which affects the probability values produced. Barbosa (2006) and Real, Barbosa & Vargas (2006) proposed an environmental favourability function which is based on logistic regression but cancels out uneven proportions of presences and absences in the modelled data. Favourability thus assesses the extent to which the environmental conditions change the probability of occurrence of a species with respect to its overall prevalence in the study area. Model predictions become, therefore, directly comparable among species with different prevalences. The favourability function is implemented in the **fuzzySim** package and is also in the SAM (Spatial Analysis in Macroecology) software (Rangel et al. 2010).

Using simulated data, Albert & Thuiller (2008) proposed a modification to the favourability function, but it requires knowing the true prevalence of the species (not just the prevalence in the studied sample), which is rarely possible in real-world modelling. Besides, this suggestion was based on the misunderstanding that the favourability function was a way to obtain the probability of occurrence when prevalence differs from 50%, which is incorrect (see Acevedo & Real 2012).

To get environmental favourability with either the Real, Barbosa & Vargas ("RBV") or the Albert & Thuiller ("AT") method, you just need to get a probabilistic model (e.g. logistic regression) from your data and then use the `Fav` function. Input data for this function are either a model object resulting from the `glm` function, or the presences-absences (1-0) of your species and the corresponding presence probability values, obtained e.g. with predict(mymodel, mydata, type = "response"). Alternatively to the presences-absences, you can provide either the sample prevalence or the numbers of presences and absences. In case you want to use the "AT" method, you also need to provide the true (absolute) prevalence of your species.

## Value

A numeric vector of the favourability values corresponding to the input probability values.

## Author(s)

A. Marcia Barbosa

## References

Acevedo P. & Real R. (2012) Favourability: concept, distinctive characteristics and potential usefulness. Naturwissenschaften 99: 515-522

Albert C.H. & Thuiller W. (2008) Favourability functions versus probability of presence: advantages and misuses. Ecography 31: 417-422.

Barbosa A.M.E. (2006) Modelacion de relaciones biogeograficas entre predadores, presas y parasitos: implicaciones para la conservacion de mamiferos en la Peninsula Iberica. PhD Thesis, University of Malaga (Spain).

Rangel T.F.L.V.B, Diniz-Filho J.A.F & Bini L.M. (2010) SAM: a comprehensive application for Spatial Analysis in Macroecology. Ecography 33: 46-50.

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

## See Also

glm, multGLM

## Examples

```
# obtain a probability model and its predictions:

data(rotif.env)

names(rotif.env)

mod <- with(rotif.env, glm(Abrigh ~ Area + Altitude + AltitudeRange +
HabitatDiversity + HumanPopulation, family = binomial))

prob <- predict(mod, data = rotif.env, type = "response")


# obtain predicted favourability in different ways:

Fav(model = mod)

Fav(obs = rotif.env$Abrigh, pred = prob)

Fav(pred = mod$fitted.values, n1n0 = c(112, 179))

Fav(pred = mod$fitted.values, sample.preval = 0.38)
```

---

# FDR *False Discovery Rate*

---

## Description

Calculate the false discovery rate (type I error) under repeated testing and determine which variables to select and to exclude from multivariate analysis.

## Usage

```
FDR(data = NULL, sp.cols = NULL, var.cols = NULL, pvalues = NULL, model.type = NULL,
family = "auto", correction = "fdr", q = 0.05, verbose = TRUE, simplif = FALSE)
```

## Arguments

| | |
|---|---|
| data | a data frame containing the response and predictor variables (one in each column). |
| sp.cols | index number of the column containing the response variable (currently implemented for only one response variable at a time). |
| var.cols | index numbers of the columns containing the predictor variables. |

`pvalues` — optionally, instead of `data`, `sp.cols` and `var.cols`, a data frame with the names of the predictor variables in the first column and their bivariate p-values (obtained elsewhere) in the second column. Example: pvalues <- data.frame(var = letters[1:5], pval = c(0.02, 0.004, 0.07, 0.03, 0.05)).

`model.type` — this argument (previously a character value, either "LM" or "GLM") is now deprecated and ignored with a warning if provided. This information is now included in argument `family` - e.g., if you want linear models (LM), you can set `family = 'gaussian'`.

`family` — The error distribution and (optionally) the link function to use (see `glm` or `family` for details). The default, `'auto'`, automatically uses 'binomial' family for response variables containing only values 0 and 1, 'poisson' for positive integer responses, and 'gaussian' otherwise.

`correction` — the correction procedure to apply to the p-values; see `p.adjust.methods` for available options and `p.adjust` for more information. The default is "fdr".

`q` — the threshold value of FDR-corrected significance above which to reject variables. Defaults to 0.05.

`verbose` — logical value indicating whether to display messages.

`simplif` — logical value indicating if simplified results should be provided (see Value).

## Details

It is common in ecology to search for statistical relationships between species' occurrence and a set of predictor variables. However, when a large number of variables is analysed (compared to the number of observations), false findings may arise due to repeated testing. Garcia (2003) recommended controlling the false discovery rate (FDR; Benjamini & Hochberg 1995) in ecological studies. The `p.adjust` R function performs this and other corrections to the significance (p) values of variables under repeated testing. The `FDR` function performs repeated regressions (either linear or binary logistic) or uses already-obtained p values for a set of variables; calculates the FDR with `p.adjust`; and shows which variables should be retained for or excluded from further multivariate analysis according to their corrected p values (see, for example, Barbosa, Real & Vargas 2009).

The FDR function uses the Benjamini & Hochberg ("BH") correction by default, but check the `p.adjust` documentation for other available methods. Input data may be the response variable (for example, the presence-absence or abundance of a species) and the predictors (a table with one independent variable in each column, with the same number of rows and in the same order as the response); there should be no missing values in the data. Model type can be either "LM" (linear model) for continuous response variables such as abundance, or "GLM" (generalized linear model) for binary responses such as presence-absence. Alternatively, you may already have performed the univariate regressions and have a set of variables and corresponding p values which you want to correct with FDR; in this case, get a table with your variables' names in the first column and their p values in the second column, and supply it as the `pvalues` argument to the `FDR` function (no need to provide response or predictors in this case).

## Value

If `simplif = TRUE`, this function returns a data frame with the variables' names as row names and 4 columns containing, respectively, their individual (bivariate) coefficients against the response, their individual AIC, p-value and adjusted p-value according to the applied `correction`. If `simplif = FALSE` (the default), the result is a list of two such data frames:

`exclude` with the variables to exclude.

`select` with the variables to select (under the given `q` value).

## Author(s)

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

Benjamini Y. & Hochberg Y. (1995) Controlling the false discovery rate: a practical and powerful approach to multiple testing. Journal of the Royal Statistical Society, Series B 57: 289-300

Garcia L.V. (2003) Controlling the false discovery rate in ecological research. Trends in Ecology and Evolution 18: 553-554

## See Also

`p.adjust`

## Examples

```
data(rotif.env)

names(rotif.env)

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

FDR(data = rotif.env, sp.cols = 18, var.cols = 5:17, simplif = TRUE)
```

---

`fuzSim` *Fuzzy similarity*

---

## Description

This function calculates fuzzy similarity, based on a fuzzy version of the binary similarity index specified in `method`, between two binary (0 or 1) or fuzzy (between 0 and 1) variables.

## Usage

```
fuzSim(x, y, method, na.rm = TRUE)
```

## Arguments

`x` — a vector of (optionally fuzzy) presence-absence data, with 1 meaning presence, 0 meaning absence, and values in between meaning fuzzy presence (or the degree to which each locality belongs to the set of species presences, or to which each species belongs to the locality; Zadeh, 1965). Fuzzy presence-absence can be obtained, for example, with functions `multGLM`, `distPres` or `multTSA` in this package.

`y` — a vector similar to `x`, of the same length and in the same order.

`method` — the similarity index to use. Currently available options are `'Jaccard'`, `'Sorensen'`, `'Simpson'` and `'Baroni'` (see Details).

`na.rm` — logical value indicating whether `NA` values should be ignored. The default is `TRUE`.

## Details

Similarity between ecological communities, beta diversity patterns, biotic regions, and distributional relationships among species are commonly determined based on pair-wise (dis)similarities in species' occurrence patterns. Some of the most commonly employed similarity indices are those of Jaccard (1901), Sorensen (1948), Simpson (1960) and Baroni-Urbani & Buser (1976), which are here implemented in their fuzzy versions (Barbosa, 2015), able to deal with both binary and fuzzy data. Jaccard's and Baroni's indices have associated tables of significant values (Baroni-Urbani & Buser 1976, Real & Vargas 1996, Real 1999).

## Value

The function returns a value between 0 and 1 representing the fuzzy similarity between `x` and `y`. Note, for example, that Jaccard similarity can be converted to dissimilarity (or Jaccard distance) if subtracted from 1, while `1-Sorensen` is not a proper distance metric as it lacks the property of triangle inequality (see `http://en.wikipedia.org/wiki/S%C3%B8rensen%E2%80%93Dice_coefficient`).

## Note

The formulas used in this function may look slighty different from some of their published versions (e.g. Baroni-Urbani & Buser 1976), not only because the letters are switched, but because here the A and B are the numbers of attributes present in each element, whether or not they are also present in the other one. Thus, our 'A+B' is equivalent to 'A+B+C' in formulas where A and B are the numbers of attributes present in one but not the other element, and our A+B-C is equivalent to their A+B+C. The formulas used here (adapted from Olivero et al. 1998) are faster to calculate, visibly for large datasets.

## Author(s)

A. Marcia Barbosa

## References

Barbosa A.M. (2015) fuzzySim: applying fuzzy logic to binary similarity indices in ecology. Methods in Ecology and Evolution, 6: 853-858.

Baroni-Urbani C. & Buser M.W. (1976) Similarity of Binary Data. Systematic Zoology, 25: 251-259

Jaccard P. (1901) Etude comparative de la distribution florale dans une portion des Alpes et des Jura. Memoires de la Societe Vaudoise des Sciences Naturelles, 37: 547-579

Olivero J., Real R. & Vargas J.M. (1998) Distribution of breeding, wintering and resident waterbirds in Europe: biotic regions and the macroclimate. Ornis Fennica, 75: 153-175

Real R. (1999) Tables of significant values of Jaccard's index of similarity. Miscellania Zoologica 22: 29:40

Real R. & Vargas J.M (1996) The probabilistic basis of Jaccard's index of similarity. Systematic Biology 45: 380-385

Simpson, G.G. (1960) Notes on the measurement of faunal resemblance. Amer. J. Sci. 258A, 300-311

Sorensen T. (1948) A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons. Kongelige Danske Videnskabernes Selskab, 5(4): 1-34

Zadeh L.A. (1965) Fuzzy sets. Information and Control, 8: 338-353

## See Also

simMat; modOverlap

## Examples

```
data(rotif.env)

names(rotif.env)


# you can calculate similarity between binary species occurrence patterns:

fuzSim(rotif.env[ , "Abrigh"], rotif.env[ , "Afissa"], method = "Jaccard")
fuzSim(rotif.env[ , "Abrigh"], rotif.env[ , "Afissa"], method = "Sorensen")
fuzSim(rotif.env[ , "Abrigh"], rotif.env[ , "Afissa"], method = "Simpson")
fuzSim(rotif.env[ , "Abrigh"], rotif.env[ , "Afissa"], method = "Baroni")


# or you can model environmental favourability for these species
# and calculate fuzzy similarity between their environmental predictions
# which goes beyond the strict coincidence of their occurrence records:

fav <- multGLM(rotif.env, sp.cols = 18:19, var.cols = 5:17, step = TRUE,
FDR = TRUE, trim = TRUE, P = FALSE, Fav = TRUE) $ predictions

fuzSim(fav[ , "Abrigh_F"], fav[ , "Afissa_F"], method = "Jaccard")
fuzSim(fav[ , "Abrigh_F"], fav[ , "Afissa_F"], method = "Sorensen")
```

```
fuzSim(fav[ , "Abrigh_F"], fav[ , "Afissa_F"], method = "Simpson")
fuzSim(fav[ , "Abrigh_F"], fav[ , "Afissa_F"], method = "Baroni")
```

---

`fuzzyOverlay` *Row-wise overlay operations based on fuzzy logic*

---

### Description

Logical and set operations are useful for comparative distribution modelling, to assess consensus or mismatches between the predictions of different models, and to quantify differences between models obtained for different time periods. Fuzzy set theory (Zadeh 1965, Barbosa & Real 2012) allows performing such operations without converting the predictions from continuous to binary, with the inherent application of arbitrary thresholds and over-simplification of model predictions. The result is a continuous numerical value quantifying the intersection, union, sum, or other operation among model predictions, whether binary or continuous.

### Usage

```
fuzzyOverlay(data, overlay.cols = 1:ncol(data), op = "intersection",
na.rm = FALSE, round.digits = 2)
```

### Arguments

| | |
|---|---|
| data | matrix or data frame containing the model predictions to compare. |
| overlay.cols | vector of the names or index numbers of the columns to compare. The default is all columns in data. |
| op | character value indicating the operation to perform between the prediction columns in data. Can be 'consensus' for the arithmetic mean of predictions (or the fuzzy equivalent of the proportion of models that agree that the species occurs at each site), 'fuzzy_and' or 'intersection' for fuzzy intersection; 'fuzzy_or' or 'union' for fuzzy union; 'prob_and' or 'prob_or' for probabilistic and/or, respectively (see Details); 'maintenance' for the values where all predictions for the same row (rounded to the number of digits specified in the next argument) are the same. If data has only two columns to compare, you can also calculate 'xor' for exclusive or, 'AnotB' for the the occurrence of the species in column 1 in detriment of that in column 2,'expansion' for the prediction increase in rows where column 2 has higher values than column 1, 'contraction' for the prediction decrease in rows where column 2 has lower values than column 1, or 'change' for a mix of the latter two, with positive values where there has been an increase and negative values where there was decrease in favourability from columns 1 to 2. For expansion, contraction and maintenance, rows where the values do not satisfy the condition (i.e. second column larger, smaller, or roughly equal to the first column) get a value of zero. |
| na.rm | logical value indicating if NA values should be ignored. The default is FALSE, so rows with NA in any of the prediction columns get NA as a result. |
| round.digits | integer value indicating the number of decimal places to be used if op = 'maintenance'. The default is 2. |

## Details

If your predictions are probabilities, 'prob_and' (probabilistic "and") gives the probability of all species in data occurring simultaneously by multiplying all probabilities; and 'prob_or' (probabilistic "or") gives the probability of any of them occurring at each site. These can be quite restrictive, though; probabilistic "and" can give particularly irrealistically small values.

If you have (or convert your probabilities to) favourability predictions, which can be used directly with fuzzy logic (Real et al. 2006; see Fav function), you can use 'fuzzy_and' or 'intersection' to get the favourability for all species co-occurring at each site, and 'fuzzy_or' or 'union' to get favourability for any of them to occur at each site (Barbosa & Real 2012).

## Value

This function returns a vector, with length equal to the number of rows in data, containing the row-wise result of the operation performed.

## Author(s)

A. Marcia Barbosa

## References

Barbosa A.M. & Real R. (2012) Applying fuzzy logic to comparative distribution modelling: a case study with two sympatric amphibians. The Scientific World Journal, 2012, Article ID 428206

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

Zadeh, L.A. (1965) Fuzzy sets. Information and Control, 8: 338-353

## See Also

fuzSim, modOverlap and fuzzyRangeChange for overall (not row-wise) comparisons among model predictions.

## Examples

```
data(rotif.env)

names(rotif.env)


# get model predictions for 3 of the species in rotif.env:

mods <- multGLM(rotif.env, sp.cols = 18:20, var.cols = 5:17, id.col = 1,
step = TRUE, FDR = TRUE, trim = TRUE)

preds <- mods$predictions[ , c("Abrigh_F", "Afissa_F", "Apriod_F")]


# calculate intersection and union among those predictions:
```

```
preds$intersect <- fuzzyOverlay(preds, op = "intersection")

preds$union <- fuzzyOverlay(preds, op = "union")

head(preds)


# imagine you have a model prediction for species 'Abrigh' in a future time
# (here we will create one by randomly jittering the current predictions)

preds$Abrigh_imag <- jitter(preds[ , "Abrigh_F"], amount = 0.2)
preds$Abrigh_imag[preds$Abrigh_imag < 0] <- 0
preds$Abrigh_imag[preds$Abrigh_imag > 1] <- 1


# you can calculate row-wise prediction changes from Abrigh to Abrigh_imag:

preds$Abrigh_exp <- fuzzyOverlay(preds, overlay.cols = c("Abrigh_F",
"Abrigh_imag"), op = "expansion")

preds$Abrigh_contr <- fuzzyOverlay(preds, overlay.cols = c("Abrigh_F",
"Abrigh_imag"), op = "contraction")

preds$Abrigh_chg <- fuzzyOverlay(preds, overlay.cols = c("Abrigh_F",
"Abrigh_imag"), op = "change")

preds$Abrigh_maint <- fuzzyOverlay(preds, overlay.cols = c("Abrigh_F",
"Abrigh_imag"), op = "maintenance")

head(preds)
```

---

`fuzzyRangeChange` *Range change based on continuous (fuzzy) values*

---

## Description

This function quantifies overall range change (fuzzy expansion, contraction, maintenance and balance) based on the continuous predictions of two models.

## Usage

```
fuzzyRangeChange(pred1, pred2, prop = TRUE, na.rm = TRUE, round.digits = 2)
```

## Arguments

| | |
|---|---|
| `pred1` | numeric vector containing the predictions (between 0 and 1) of the model that will serve as reference. |
| `pred2` | numeric vector containing the predictions (between 0 and 1) of the model whose change will be calculated. Must be of the same length and in the same order as `pred1`. |

`prop` logical value indicating whether results should be given proportionally to the total number of cases.

`na.rm` logical value indicating whether NA values should be ignored. The default is TRUE.

`round.digits` argument to pass to `fuzzyOverlay`, indicating the number of decimal places to which to round `pred` for calculating 'maintenance' or 'stability'. The default is 2.

## Value

This function returns a data frame with the following values in different rows:

`Gain` sum of the predicted values that have increased from `pred1` to `pred2` (fuzzy equivalent of the number of gained presences)

`Loss` sum of the predicted values that have decreased from `pred1` to `pred2` (fuzzy equivalent of the number of lost presences)

`Stable_presence` fuzzy equivalent of the number of predicted presences that have remained as such (when rounded to `round.digits`) between `pred1` and `pred2`

`Stable_absence` fuzzy equivalent of the number of predicted absences that have remained as such (when rounded to `round.digits`) between `pred1` and `pred2`)

`Balance` sum of the change in predicted values from `pred1` to `pred2` (fuzzy equivalent of the balance of gained and lost presences)

All these values are divided by the total number of reference values (i.e., the fuzzy range or non-range size) if `prop = TRUE` (the default).

## Author(s)

A. Marcia Barbosa

## See Also

`fuzSim`, `modOverlap` for other ways to compare models; `fuzzyOverlay` for row-wise model comparisons

## Examples

```
# get an environmental favourability model for a rotifer species:

data(rotif.env)

names(rotif.env)

fav_current <- multGLM(rotif.env, sp.cols = 18, var.cols = 5:17, step = TRUE,
FDR = TRUE, trim = TRUE, P = FALSE, Fav = TRUE) $ predictions


# imagine you have a model prediction for this species in a future time
```

```
# (here we will create one by randomly jittering the current predictions)

fav_imag <- jitter(fav_current, amount = 0.2)
fav_imag[fav_imag < 0] <- 0
fav_imag[fav_imag > 1] <- 1


# calculate range change given by current and imaginary future predictions:

fuzzyRangeChange(fav_current, fav_imag)

fuzzyRangeChange(fav_current, fav_imag, prop = FALSE)
```

---

## getPreds *Get model predictions*

---

### Description

This function allows getting the predictions of multiple models when applied to a given dataset. It can be useful if you have a list of model objects (e.g. resulting from `multGLM`) and want to apply them to a new data set containing the same variables for another region or time period. There are options to include the logit link (`Y`) and/or `Favourability` (see `Fav`).

### Usage

```
getPreds(data, models, id.col = NULL, Y = FALSE, P = TRUE, Favourability = TRUE,
incl.input = FALSE)
```

### Arguments

| | |
|---|---|
| `data` | the data frame to which to apply the `models` to get their predictions; must contain all variables (with the same names, case-sensitive) included in any of the `models`. |
| `models` | a list of model objects obtained e.g. with function `glm` or `multGLM`. |
| `id.col` | optionally, the index number of a column of `data` containing row identifiers, to be included in the result. Ignored if `incl.input = TRUE`. |
| `Y` | logical, whether to include the logit link (y) value in the predictions. |
| `P` | logical, whether to include the probability value in the predictions. |
| `Favourability` | logical, whether to include Favourability in the predictions (see `Fav`). |
| `incl.input` | logical, whether to include input columns in the output. The default is FALSE. |

### Value

This function returns a data frame containing the model predictions, next to the `id.col` if provided, and to the input data if `incl.input = TRUE`.

## Author(s)

A. Marcia Barbosa

## See Also

multGLM, predict

## Examples

```
data(rotif.env)

names(rotif.env)


# identify rotifer data corresponding to the Eastern and Western hemispheres:

unique(rotif.env$CONTINENT)

rotif.env$HEMISPHERE <- "Eastern"

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

head(rotif.env)


# separate the rotifer data into hemispheres

east.hem <- rotif.env[rotif.env$HEMISPHERE == "Eastern", ]
west.hem <- rotif.env[rotif.env$HEMISPHERE == "Western", ]


# make models for 3 of the species in rotif.env based on their distribution
# in the Eastern hemisphere:

mods <- multGLM(east.hem, sp.cols = 18:20, var.cols = 5:17, id.col = 1,
step = FALSE, FDR = FALSE, trim = FALSE)


# get the models' predictions for the Western hemisphere dataset:

preds <- getPreds(west.hem, models = mods$models, P = TRUE, Favourability = TRUE)

head(preds)
```

---

integerCols    *Classify integer columns*

---

## Description

This function detects which numeric columns in a data frame contain only whole numbers, and converts those columns to integer class, so that they take up less space.

## Usage

```
integerCols(data)
```

## Arguments

| | |
|---|---|
| `data` | a data frame containing possibly integer columns classified as `numeric`. |

## Value

The function returns a data frame with the same columns as `data`, but with those that are numeric and contain only whole numbers (possibly including NA) now classified as `integer`.

## Author(s)

A. Marcia Barbosa

## See Also

`is.integer`, `as.integer`, `multConvert`

## Examples

```
dat <- data.frame(
  var1 = 1:10,
  var2 = as.numeric(1:10),
  var3 = as.numeric(c(1:4, NA, 6:10)),
  var4 = as.numeric(c(1:3, NaN, 5, Inf, 7, -Inf, 9:10)),
  var5 = as.character(1:10),
  var6 = seq(0.1, 1, by = 0.1),
  var7 = letters[1:10]
)  # creates a sample data frame

dat

str(dat)
# var2 classified as 'numeric' but contains only whole numbers
# var3 same as var2 but containing also NA values
# var4 same as var2 but containing also NaN and infinite values
# var5 contains only whole numbers but initially classified as factor

dat <- integerCols(dat)

str(dat)
# var2 and var3 now classified as 'integer'
# var4 remains as numeric because contains infinite and NaN (not integer) values
# var5 remains as factor
```

`modelTrim` *Trim off non-significant variables from a model*

## Description

This function performs a stepwise removal of non-significant variables from a model.

## Usage

```
modelTrim(model, method = "summary", alpha = 0.05)
```

## Arguments

| | |
|---|---|
| `model` | a model object. |
| `method` | the method for getting the individual p-values. Can be either "summary" for the p-values of the coefficient estimates, or "anova" for the p-values of the variables themselves (see Details). |
| `alpha` | the p-value above which a variable is removed. |

## Details

Stepwise variable selection is a common procedure for simplifying models. It maximizes predictive efficiency in an objective and reproducible way, and is useful when the individual importance of the predictors is not known a priori (Hosmer & Lemeshow, 2000). The `step` R function performs such procedure using an information criterion (AIC) to select the variables, but it often leaves variables that are not significant in the model. Such variables can be subsequently removed with a manual stepwise procedure (e.g. Crawley 2007, p. 442; Barbosa & Real 2010, 2012; Estrada & Arroyo 2012). The `modelTrim` function performs such removal automatically until all remaining variables are significant. It can also be applied to a full model (i.e., without previous use of the step function), as it serves as a backward stepwise selection procedure based on the significance of the coefficients (if `method = "summary"`, the default) or on the significance of the variables themselves (if `method = "anova"`, better when there are categorical variables in the model).

## Value

The input model object after removal of non-significant variables.

## Author(s)

A. Marcia Barbosa

## References

Barbosa A.M. & Real R. (2010) Favourable areas for expansion and reintroduction of Iberian lynx accounting for distribution trends and genetic diversity of the European rabbit. Wildlife Biology in Practice 6: 34-47

Barbosa A.M. & Real R. (2012) Applying fuzzy logic to comparative distribution modelling: a case study with two sympatric amphibians. The Scientific World Journal, Article ID 428206

Crawley M.J. (2007) The R Book. John Wiley & Sons, Chichester (UK)

Estrada A. & Arroyo B. (2012) Occurrence vs abundance models: Differences between species with varying aggregation patterns. Biological Conservation, 152: 37-45

Hosmer D. W. & Lemeshow S. (2000) Applied Logistic Regression (2nd ed). John Wiley and Sons, New York

## See Also

step

## Examples

```
# load sample data:

data(rotif.env)

names(rotif.env)


# build a stepwise model of a species' occurrence based on some of the variables:

mod <- with(rotif.env, step(glm(Abrigh ~ Area + Altitude + AltitudeRange +
HabitatDiversity + HumanPopulation, family = binomial)))


# examine the model:

summary(mod)  # contains non-significant variables


# use modelTrim to get rid of non-significan effects:

mod <- modelTrim(mod)

summary(mod)  # only significant variables now
```

---

# modOverlap *Overall overlap between model predictions*

---

## Description

This function calculates the degree of overlap between the predictions of two models, using niche comparison metrics such as Schoener's D, Hellinger distance and Warren's I.

## Usage

```
modOverlap(pred1, pred2, na.rm = TRUE)
```

## Arguments

| | |
|---|---|
| `pred1` | numeric vector of the predictions of a generalized linear model (values between 0 and 1). |
| `pred2` | numeric vector of the predictions of another generalized linear model; must be of the same length and in the same order as `pred1`. |
| `na.rm` | logical value indicating whether `NA` values should be removed prior to calculation. |

## Details

See Warren et al. (2008).

## Value

This function returns a list of 3 metrics:

| | |
|---|---|
| `SchoenerD` | Schoener's (1968) D statistic for niche overlap, varying between 0 (no overlap) and 1 (identical niches). |
| `WarrenI` | the I index of Warren et al. (2008), based on Hellinger distance (below) but re-formulated to also vary between 0 (no overlap) and 1 (identical niches). |
| `HellingerDist` | Hellinger distance (as in van der Vaart 1998, p. 211) between probability distributions, varying between 0 and 2. |

## Note

A function providing similar measures, `niche.overlap`, is available in package **phyloclim**, but it requires complex and software-specific input data formats.

## Author(s)

A. Marcia Barbosa

## References

Schoener T.W. (1968) Anolis lizards of Bimini: resource partitioning in a complex fauna. Ecology 49: 704–726

van der Vaart A.W. (1998) Asymptotic statistics. Cambridge Univ. Press, Cambridge (UK)

Warren D.L., Glor R.E. & Turelli M. (2008) Environmental niche equivalency versus conservatism: quantitative approaches to niche evolution. Evolution, 62: 2868–83 (and further ERRATUM)

## See Also

`fuzSim`; `niche.overlap` in package **phyloclim**

## Examples

```
# get an environmental favourability model for a rotifer species:

data(rotif.env)

names(rotif.env)

fav_current <- multGLM(rotif.env, sp.cols = 18, var.cols = 5:17, step = TRUE,
FDR = TRUE, trim = TRUE, P = FALSE, Fav = TRUE) $ predictions


# imagine you have a model prediction for this species in a future time
# (here we will create one by randomly jittering the current predictions)

fav_imag <- jitter(fav_current, amount = 0.2)
fav_imag[fav_imag < 0] <- 0
fav_imag[fav_imag > 1] <- 1


# calculate niche overlap between current and imaginary future predictions:

modOverlap(fav_current, fav_imag)
```

---

# multConvert *Multiple conversion*

---

## Description

This function can simultaneously convert multiple columns of a matrix or data frame.

## Usage

```
multConvert(data, conversion, cols = 1:ncol(data))
```

## Arguments

| | |
|---|---|
| `data` | A matrix or data frame containing columns that need to be converted |
| `conversion` | the conversion to apply, e.g. `as.factor` or a custom-made fucntion |
| `cols` | the columns of `data` to convert |

## Details

Sometimes we need to change the data type (class, mode) of a variable in R. There are various possible conversions, performed by functions like as.integer, as.factor or as.character. If we need to perform the same conversion on a number of variables (columns) in a data frame, we can convert them all simultaneously using this function. By default it converts all the columns in the data frame, but you can specify just a few of them. multConvert can also be used to apply other kinds of transformations - for example, if you need to divide some of your columns by 100, just write a function to do this and then use multConvert to apply this function to any group of columns.

## Value

The input data with the specified columns converted as asked.

## Author(s)

A. Marcia Barbosa

## Examples

```
data(rotif.env)

str(rotif.env)

# convert the first 4 columns to character:
converted.rotif.env <- multConvert(data = rotif.env, conversion = as.character, cols = 1:4)

str(converted.rotif.env)


names(rotif.env)

# divide some columns by 100:

div100 <- function(x) x / 100

rotif.env.cent <- multConvert(data = rotif.env, conversion = div100, cols = c(6:10, 12:17))

head(rotif.env.cent)
```

---

`multGLM` *GLMs for multiple species with multiple options*

---

## Description

This function calculates generalized linear models for a set of (species) presence/absence records in a data frame, with a wide set of options for data partition, variable selection, and output form.

## Usage

```
multGLM(data, sp.cols, var.cols, id.col = NULL, family = "binomial",
test.sample = 0, FDR = FALSE, corSelect = FALSE, cor.thresh = 0.8, step = TRUE,
trace = 0, start = "null.model", direction = "both", Y.prediction = FALSE,
P.prediction = TRUE, Favourability = TRUE, group.preds = TRUE, trim = TRUE, ...)
```

## Arguments

| | |
|---|---|
| `data` | a data frame in wide format (see `splist2presabs`) containing, in separate columns, your species' binary (0/1) occurrence data and the predictor variables. |
| `sp.cols` | index numbers of the columns containing the species data to be modelled. |
| `var.cols` | index numbers of the columns containing the predictor variables to be used for modelling. |
| `id.col` | (optional) index number of column containing the row identifiers (if defined, it will be included in the output `predictions` data frame). |
| `family` | argument to be passed to the `glm` function; only 'binomial' is implemented in `multGLM` so far. |
| `test.sample` | a subset of data to set aside for subsequent model testing. Can be a value between 0 and 1 for a proportion of the data to choose randomly (e.g. 0.2 for 20%), or an integer number for a particular number of cases to choose randomly among the records in `data`, or a vector of integers for the index numbers of the particular rows to set aside, or "Huberty" for his rule of thumb based on the number of variables (Huberty 1994, Fielding & Bell 1997). |
| `FDR` | logical value indicating whether to do a preliminary exclusion of variables based on the false discovery rate (see `FDR`). The default is `FALSE`. |
| `corSelect` | logical value indicating whether to do a preliminary exclusion of highly correlated variables (see `corSelect`). The default is `FALSE`. |
| `cor.thresh` | numerical value indicating the correlation threshold to pass to `corSelect` (used only if `corSelect = TRUE`). |
| `step` | logical, whether to use the `step` function to perform a stepwise variable selection based on AIC. |
| `trace` | if positive, information is printed during the running of `step`. Larger values may give more detailed information. |
| `start` | character, whether to start with the 'null.model' (so AIC variable selection starts forward) or with the 'full.model' (so selection starts backward). Used only if `step = TRUE`. |
| `direction` | argument to be passed to `step` specifying the direction of variable selection ('forward', 'backward' or 'both'). Used only if `step = TRUE`. |
| `Y.prediction` | logical, whether to include output predictions in the scale of the predictor variables (`type = "link"` in `predict.glm`). |
| `P.prediction` | logical, whether to include output predictions in the scale of the response variable, i.e. probability (`type = "response"` in `predict.glm`). |
| `Favourability` | logical, whether to apply the `Fav`ourability function to extract the effect of prevalence on probability (Real et al. 2006) and include its results in the output. |
| `group.preds` | logical, whether to group together predictions of similar type (`Y`, `P` or `F`) in the output `predictions` table (e.g. if `FALSE`: sp1_Y, sp1_P, sp1_F, sp2_Y, sp2_P, sp2_F; if `TRUE`: sp1_Y, , sp2_Y, sp1_P, sp2_P, sp1_F, sp2_F). |
| `trim` | logical, whether to trim non-significant variables off the models using the `modelTrim` function; can be used whether or not `step` is TRUE; works as a backward variable elimination procedure based on significance. |
| `...` | additional arguments to be passed to `modelTrim`. |

## Details

This function automatically calculates binomial GLMs for one or more species (or other binary variables) in a data frame. The function can optionally perform `step`wise variable selection (and it does so by default) instead of forcing all variables into the models, starting from either the null model (the default, so selection starts forward) or from the full model (so selection starts backward) and using Akaike's information criterion (AIC) as a variable selection criterion. Instead or subsequently, it can also perform stepwise removal of non-significant variables from the models using the `modelTrim` function.

There is also an optional preliminary selection of non-correlated variables, and/or of variables with a significant bivariate relationship with the response, based on the false discovery rate (`FDR`). Note, however, that some variables can be significant in a multivariate model even if they would not have been selected by FDR.

`Fav`ourability is also calculated, removing the effect of species prevalence from occurrence probability and thus allowing direct comparisons between models (Real et al. 2006).

By default, all data are used in model training, but you can define an optional `test.sample` to be reserved for model testing afterwards. You may also want to do a previous check for `multicol`linearity among variables, e.g. the variance inflation factor (VIF).

The `multGLM` function will create a list of the resulting models (each with the name of the corresponding species column) and a data frame with their predictions (Y, P and/or F, all of which are optional). If you plan on representing these predictions in a GIS based on .dbf tables, remember that dbf only allows up to 10 characters in column names; `multGLM` predictions will add 2 characters (_Y, _P and/or _F) to each of your species column names, so use species names/codes with up to 8 characters in the data set that you are modelling. You can create (sub)species name abbreviations with the `spCodes` function.

## Value

This function returns a list with the following components:

`predictions` a data frame with the model predictions (if either of `Y.prediction`, `P.prediction`, or `Favourability` are `TRUE`).

`models` a list of the resulting model objects.

## Author(s)

A. Marcia Barbosa

## References

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

Huberty C.J. (1994) Applied Discriminant Analysis. Wiley, New York, 466 pp. Schaafsma W. & van Vark G.N. (1979) Classification and discrimination problems with applications. Part IIa. Statistica Neerlandica 33: 91-126

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

## See Also

glm, Fav, step, modelTrim, multicol, corSelect

## Examples

```
data(rotif.env)

names(rotif.env)


# make models for 3 of the species in rotif.env:

mods <- multGLM(rotif.env, sp.cols = 45:47, var.cols = 5:17, id.col = 1,
step = TRUE, FDR = TRUE, trim = TRUE)

names(mods)

head(mods$predictions)

names(mods$models)

mods$models[[1]]

mods$models[["Ttetra"]]
```

---

multicol *Analyse multicollinearity in a dataset, including VIF*

---

## Description

This function analyses multicollinearity in a set of variables or in a model, including the R-squared, tolerance and variance inflation factor (VIF).

## Usage

```
multicol(vars = NULL, model = NULL, reorder = TRUE)
```

## Arguments

| | |
|---|---|
| vars | A matrix or data frame containing the numeric variables for which to calculate multicollinearity. Only the 'independent' (predictor, explanatory, right hand side) variables should be entered, as the result obtained for each variable depends on all the other variables present in the analysed data set. |
| model | Alternatively to vars, a glm model object to calculate multicol among the included variables. |
| reorder | logical, whether variables should be output in decreasing order or VIF value rather than in their input order. The default is TRUE. |

## Details

Testing collinearity among covariates is a recommended step of data exploration before applying a statistical model (Zuur et al. 2010). However, you can also calculate multicollinearity among the variables already included in a model.

The multicol function calculates the degree of multicollinearity in a set of numeric variables, using three closely related measures: R squared (the coefficient of determination of a linear regression of each predictor variable on all other predictor variables, i.e., the amount of variation in each variable that is accounted for by other variables in the dataset); tolerance (1 - R squared), i.e. the amount of variation in each variable that is not included in the remaining variables; and the variance inflation factor: VIF = 1 / (1 - R squared), which, in a linear model with these variables as predictors, reflects the degree to which the variance of an estimated regression coefficient is increased due only to the correlations among covariates (Marquardt 1970; Mansfield & Helms 1982).

## Value

The function returns a matrix with one row per analysed variable, the names of the variables as row names, and 3 columns: R-squared, Tolerance and VIF.

## Author(s)

A. Marcia Barbosa

## References

Marquardt D.W. (1970) Generalized inverses, ridge regression, biased linear estimation, and non-linear estimation. Technometrics 12: 591-612.

Mansfield E.R. & Helms B.P. (1982) Detecting multicollinearity. The American Statistician 36: 158-160.

Zuur A.F., Ieno E.N. & Elphick C.S. (2010) A protocol for data exploration to avoid common statistical problems. Methods in Ecology and Evolution 1: 3-14.

## Examples

```
data(rotif.env)
names(rotif.env)

# calculate multicollinearity among the predictor variables:
multicol(rotif.env[ , 5:17], reorder = FALSE)
multicol(rotif.env[ , 5:17])

# you can also calculate multicol among the variables included in a model:
mod <- step(glm(Abrigh ~ Area + Altitude + AltitudeRange + HabitatDiversity +
HumanPopulation + Latitude + Longitude + Precipitation + PrecipitationSeasonality +
TemperatureAnnualRange + Temperature + TemperatureSeasonality + UrbanArea,
data = rotif.env))
multicol(model = mod)

# more examples using R datasets:
multicol(trees)
```

```
# you'll get a warning and some NA results if any of the variables is not numeric:
multicol(OrchardSprays)

# so define the subset of numeric 'vars' to calculate 'multicol' for:
multicol(OrchardSprays[ , 1:3])
```

---

multTSA *Trend Surface Analysis for multiple species*

---

## Description

This function performs trend surface analysis for multiple species at a time. It converts categorical presence-absence (1-0) data into continuous surfaces denoting the spatial trend in species' occurrence patterns.

## Usage

```
multTSA(data, sp.cols, coord.cols, id.col = NULL, degree = 3, step = TRUE,
Favourability = FALSE, suffix = "_TS", save.models = FALSE)
```

## Arguments

| | |
|---|---|
| `data` | a matrix or data frame containing, at least, two columns with spatial coordinates, and one column per species containing their presence (1) and absence (0) data, with localities in rows. |
| `sp.cols` | names or index numbers of the columns containing the species presences and absences in data. Must contain only zeros (0) for absences and ones (1) for presences. |
| `coord.cols` | names or index numbers of the columns containing the spatial coordinates in data (x and y, or longitude and latitude, in this order!). |
| `id.col` | optionally, the name or index number of a column (to be included in the output) containing locality identifiers in data. |
| `degree` | the degree of the spatial polynomial to use (see Details). The default is 3. |
| `step` | logical value indicating whether the regression of presence-absence on the spatial polynomial should do a stepwise inclusion of the polynomial terms (using the `step` function with default settings, namely backward AIC selection), rather than forcing all terms into the equation. The default is `TRUE`. |
| `Favourability` | logical value indicating whether the probability values obtained from the regression should be converted to favourability, so that they are more directly comparable among species with different prevalence (see Real et al. 2006, Acevedo & Real 2012). The default is `FALSE`. |
| `suffix` | character indicating the suffix to add to the trend surface columns in the resulting data frame. The default is `"_TS"`. |
| `save.models` | logical value indicating whether the models obtained from the regressions should be saved in the results. The default is `FALSE`. |

## Details

Trend Surface Analysis is a way to model the spatial structure in species' distributions by regressing occurrence data on the spatial coordinates x and y, for a linear trend, or on polynomial terms of these coordinates (x^2, y^2, x*y, etc.), for curvilinear trends (Legendre & Legendre, 1998; Borcard et al., 2011). Second- and third-degree polynomials are often used. `multTSA` allows specifying the degree of the spatial polynomial to use. By default, it uses a 3rd-degree polynomial and performs stepwise AIC selection of the polynomial terms to include.

## Value

A matrix or data frame containing the identifier column (if provided in `id.col`) and one column per species containing the value predicted by the trend surface analysis.

## Author(s)

A. Marcia Barbosa

## References

Acevedo P. & Real R. (2012) Favourability: concept, distinctive characteristics and potential usefulness. Naturwissenschaften 99: 515-522

Borcard D., Gillet F. & Legendre P. (2011) Numerical Ecology with R. Springer, New York.

Legendre P. & Legendre L. (1998) Numerical Ecology. Elsevier, Amsterdam.

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

## See Also

`distPres`, `poly`

## Examples

```
data(rotif.env)

head(rotif.env)

names(rotif.env)

tsa <- multTSA(rotif.env, sp.cols = 18:47, coord.cols = c("Longitude", "Latitude"),
id.col = 1)

head(tsa)
```

## percentTestData *Percent test data*

### Description

Based on the work of Schaafsma & van Vark (1979), Huberty (1994) provided a heuristic ("rule of thumb") for determining an adequate proportion of data to set aside for testing species presence/absence models, based on the number of predictor variables that are used (Fielding & Bell 1997). The `percentTestData` function calculates this proportion as a percentage.

### Usage

```
percentTestData(nvar)
```

### Arguments

| | |
|---|---|
| `nvar` | the number of variables in the model. |

### Value

A numeric value of the percentage of data to leave out of the model for further model testing.

### Author(s)

A. Marcia Barbosa

### References

Huberty C.J. (1994) Applied Discriminant Analysis. Wiley, New York, 466 pp.

Schaafsma W. & van Vark G.N. (1979) Classification and discrimination problems with applications. Part IIa. Statistica Neerlandica 33: 91-126

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

### See Also

`multGLM`

### Examples

```
# say you're building a model with 15 variables:

percentTestData(15)

# the result tells you that 21% is an appropriate percentage of data
# to set aside for testing your model, so train it with 79% of the data
```

rotif.env — *Rotifers and environmental variables on TDWG level 4 regions of the world*

## Description

These data were extracted from a database of monogonont rotifer species presence records on the geographical units used by the Biodiversity Information Standards (formerly Taxonomic Database Working Group, TDWG; base maps available from www.kew.org/science-conservation/research-data/resources/gis-unit/tdwg-world) and a few environmental (including human and spatial) variables on the same spatial units. The original data were compiled and published by Fontaneto et al. (2012) in long (narrow, stacked) format. Here they are presented in wide or unstacked format (presence-absence table, obtained with the splist2presabs function), reduced to the species recorded in at least 100 (roughly one third) different TDWG level 4 units, and with abbreviations of the species' names (obtained with the spCodes function). Mind that this is not a complete picture of these species' distributions, due to insufficient sampling in many regions.

## Usage

```
data(rotif.env)
```

## Format

A data frame with 291 observations on the following 47 variables.

`TDWG4` a factor with 291 levels indicating the abbreviation code of each TDWG4 region

`LEVEL_NAME` a factor with 291 levels indicating the name of each TDWG4 region

`REGION_NAME` a factor with 47 levels indicating the name of the main geographical region to which each TDWG4 level belongs

`CONTINENT` a factor with 9 levels indicating the continent to which each TDWG4 level belongs

`Area` a numeric vector

`Altitude` a numeric vector

`AltitudeRange` a numeric vector

`HabitatDiversity` a numeric vector

`HumanPopulation` a numeric vector

`Latitude` a numeric vector

`Longitude` a numeric vector

`Precipitation` a numeric vector

`PrecipitationSeasonality` a numeric vector

`TemperatureAnnualRange` a numeric vector

`Temperature` a numeric vector

`TemperatureSeasonality` a numeric vector

UrbanArea a numeric vector

Abrigh a numeric vector

Afissa a numeric vector

Apriod a numeric vector

Bangul a numeric vector

Bcalyc a numeric vector

Bplica a numeric vector

Bquadr a numeric vector

Burceo a numeric vector

Cgibba a numeric vector

Edilat a numeric vector

Flongi a numeric vector

Kcochl a numeric vector

Kquadr a numeric vector

Ktropi a numeric vector

Lbulla a numeric vector

Lclost a numeric vector

Lhamat a numeric vector

Lluna a numeric vector

Llunar a numeric vector

Lovali a numeric vector

Lpatel a numeric vector

Lquadr a numeric vector

Mventr a numeric vector

Ppatul a numeric vector

Pquadr a numeric vector

Pvulga a numeric vector

Specti a numeric vector

Tpatin a numeric vector

Tsimil a numeric vector

Ttetra a numeric vector

## Source

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.env)

head(rotif.env)
```

---

`rotifers` *Rotifer species on TDWG level 4 regions of the world*

---

## Description

These data were extracted from a database of monogonont rotifer species records on the geographical units used by the Biodiversity Information Standards (formerly Taxonomic Database Working Group, TDWG; base maps available at `www.kew.org/science-conservation/research-data/resources/gis-unit/tdwg-world`). The original data were compiled and published by Fontaneto et al. (2012) for all TDWG levels. Here they are reduced to the TDWG - level 4 units and to the species recorded in at least 100 (roughly one third) of these units. Mind that this is not a complete picture of these species' distributions, due to insufficient sampling in many regions.

## Usage

```
data("rotifers")
```

## Format

A data frame with 3865 observations on the following 2 variables.

`TDWG4` a factor with 274 levels corresponding to the code names of the TDWG level 4 regions in which the records were taken

`species` a factor with 30 levels corresponding to the names of the (sub)species recorded in at least 100 different TDWG level 4 regions

## Source

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(rotifers)

head(rotifers, 10)
```

---

`simFromSetOps` *Calculate similarity from set operations*

---

## Description

This function calculates pair-wise similarity based on the results of set operations (intersection, union) among the subjects.

### Usage

```
simFromSetOps(size1, size2, intersection, union, total.size = NULL,
method = c("Jaccard", "Sorensen", "Simpson", "Baroni"), verbosity = 1)
```

### Arguments

| | |
|---|---|
| `size1` | size of subject 1 (e.g., area of the distribution range of a species, or its number of presences within a grid). Not needed if `method = "Jaccard"`. |
| `size2` | the same for subject 2. |
| `intersection` | size of the intersection among subjects 1 and 2 (area of the intersection among their distribution ranges, or number of grid cells in which they co-occur). |
| `union` | size of the union of subjects 1 and 2. |
| `total.size` | total size of the study area. Needed only when calculating a similarity index that takes shared absences into account (i.e., `method = "Baroni"`). |
| `method` | the similarity index to use. Currently implemented options are `'Jaccard'`, `'Sorensen'`, `'Simpson'` and `'Baroni'`. |
| `verbosity` | integer indicating whether to display messages. |

### Details

Similarities among ecological communities, beta diversity patterns, biotic regions, and distributional relationships among species are commonly determined based on pair-wise (dis)similarities in species' occurrence patterns. This function implements some of the most commonly employed similarity indices, namely those of Jaccard (1901), Sorensen (1948), Simpson (1960) and Baroni-Urbani & Buser (1976), based on the amount of occupied and overlap area between two species.

### Value

The numeric value of similarity among subjects 1 and 2.

### Author(s)

A. Marcia Barbosa

### References

Baroni-Urbani C. & Buser M.W. (1976) Similarity of Binary Data. Systematic Zoology, 25: 251-259

Jaccard P. (1901) Etude comparative de la distribution florale dans une portion des Alpes et des Jura. Memoires de la Societe Vaudoise des Sciences Naturelles, 37: 547-579

Simpson, G.G. (1960) Notes on the measurement of faunal resemblance. Amer. J. Sci. 258A, 300-311

Sorensen T. (1948) A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons. Kongelige Danske Videnskabernes Selskab, 5(4): 1-34

## See Also

fuzSim, simMat

## Examples

```
# take two species which occur in 22 and 35 area units, respectively
# and which overlap in 8 of those units:

sp1 <- 22
sp2 <- 35
int <- 8
uni <- sp1 + sp2 - int


# calculate similarity between their distributions based on different indices:

simFromSetOps(intersection = int, union = uni, method = "Jaccard")

simFromSetOps(sp1, sp2, int, uni, method = "Sorensen")

simFromSetOps(sp1, sp2, int, uni, method = "Simpson")


# if you want Baroni-Urbani & Buser's index
# you need to provide also the total size of your study area:

simFromSetOps(sp1, sp2, int, uni, total = 100, method = "Baroni")
```

---

# simMat — *Pair-wise (fuzzy) similarity matrix*

---

## Description

simMat takes a matrix or data frame containing species occurrence data or regional species composition, either categorical (0 or 1) or fuzzy (between 0 and 1), and uses the fuzSim function to calculate a square matrix of pair-wise similarities between them, using a fuzzy logic version (Barbosa, 2015) of the specified similarity index.

## Usage

```
simMat(data, method, diag = TRUE, upper = TRUE)
```

## Arguments

data — a matrix or data frame containing (optionally fuzzy) species presence-absence data (in wide format, i.e. one column per species), with 1 meaning presence, 0 meaning absence, and values in between for fuzzy presence (or the degree to which each locality belongs to the set of species presences; see Zadeh, 1965). Fuzzy presence-absence can be obtained, for example, with multGLM, distPres or multTSA. These data can also be transposed for comparing regional species compositions.

| | |
|---|---|
| method | the similarity index whose fuzzy version to use. See fuzSim for available options. |
| diag | logical value indicating whether the diagonal of the matrix should be filled (with ones). Defaults to TRUE. |
| upper | logical value indicating whether the upper triangle of the matrix (symmetric to the lower triangle) should be filled. Defaults to TRUE. |

## Details

The fuzzy versions of species occurrence data and of binary similarity indices introduce tolerance for small spatial differences in species' occurrence localities, allow for uncertainty about species occurrence, and may compensate for under-sampling and geo-referencing errors (Barbosa, 2015).

## Value

This function returns a square matrix of pair-wise similarities among the species distributions (columns) in `data`. Similarity is calculated with the fuzzy version of the index specified in `method`, which yields traditional binary similarity if the data are binary (0 or 1), or fuzzy similarity if the data are fuzzy (between 0 and 1) (Barbosa, 2015).

## Author(s)

A. Marcia Barbosa

## References

Barbosa A.M. (2015) fuzzySim: applying fuzzy logic to binary similarity indices in ecology. Methods in Ecology and Evolution, 6: 853-858.

## See Also

fuzSim

## Examples

```
# load and look at the rotif.env presence-absence data:

data(rotif.env)

head(rotif.env)

names(rotif.env)


# build a matrix of similarity among these binary data
# using e.g. Jaccard's index:
```

```
bin.sim.mat <- simMat(rotif.env[ , 18:47], method = "Jaccard")

head(bin.sim.mat)


# calculate a fuzzy version of the presence-absence data
# based on inverse distance to presences:

rotifers.invd <- distPres(rotif.env, sp.cols = 18:47,
coord.cols = c("Longitude", "Latitude"), id.col = 1, suffix = ".d", p = 1, inv = TRUE)

head(rotifers.invd)


# build a matrix of fuzzy similarity among these fuzzy distribution data,
# using the fuzzy version of Jaccard's index:

fuz.sim.mat <- simMat(rotifers.invd[ , -1], method = "Jaccard")

head(fuz.sim.mat)


# plot the similarity matrices as colours:

image(x = 1:ncol(bin.sim.mat), y = 1:nrow(bin.sim.mat), z = bin.sim.mat,
col = rev(heat.colors(256)), xlab = "", ylab = "", axes = FALSE,
main = "Binary similarity")
axis(side = 1, at = 1:ncol(bin.sim.mat), tick = FALSE,
labels = colnames(bin.sim.mat), las = 2)
axis(side = 2, at = 1:nrow(bin.sim.mat), tick = FALSE,
labels = rownames(bin.sim.mat), las = 2)

image(x = 1:ncol(fuz.sim.mat), y = 1:nrow(fuz.sim.mat), z = fuz.sim.mat,
col = rev(heat.colors(256)), xlab = "", ylab = "", axes = FALSE,
main = "Fuzzy similarity")
axis(side = 1, at = 1:ncol(fuz.sim.mat), tick = FALSE,
labels = colnames(fuz.sim.mat), las = 2, cex = 0.5)
axis(side = 2, at = 1:nrow(fuz.sim.mat), tick = FALSE,
labels = rownames(fuz.sim.mat), las = 2)


# plot a UPGMA dendrogram from each similarity matrix:

plot(hclust(as.dist(1 - bin.sim.mat), method = "average"),
main = "Binary cluster dendrogram")

plot(hclust(as.dist(1 - fuz.sim.mat), method = "average"),
main = "Fuzzy cluster dendrogram")


# you can get fuzzy chorotypes from these similarity matrices
# (or fuzzy biotic regions if your \code{\link{transpose}} \code{data},
# so that localities are in columns and species in rows)
```

```
# using the \pkg{RMACOQUI} package (Olivero et al. 2011)
```

---

spCodes *Obtain unique abbreviations of species names*

---

## Description

This function takes a vector of species names and converts them to abbreviated species codes containing the specified numbers of characters from the genus, the specific and optionally also the subspecific name. Separators can be specified by the user. The function checks that the resulting codes are unique.

## Usage

```
spCodes(species, nchar.gen = 3, nchar.sp = 3, nchar.ssp = 0, sep.species = " ",
sep.spcode = "")
```

## Arguments

| | |
|---|---|
| `species` | a character vector containig the species names to be abbreviated. |
| `nchar.gen` | the number of characters from the genus name to be included in the resulting species code. |
| `nchar.sp` | the number of characters from the specific name to be included in the resulting species code. |
| `nchar.ssp` | optionally, the number of characters from the subspecific name to be included in the resulting species code. Set it to 0 if you have subspecific names in `species` but do not want them included in the resulting species codes. |
| `sep.species` | the character that separates genus, specific and subspecific names in `species`. The default is a white space. |
| `sep.spcode` | the character you want separating genus and species abbreviations in the resulting species codes. The default is an empty character (no separator). |

## Value

This function returns a character vector containing the species codes resulting from the abbreviation. If the numbers of characters specified do not make for unique codes, an error message is displayed showing which `species` names caused it, so that you can try again with different `nchar.gen`, `nchar.sp` and/or `nchar.ssp`.

## Author(s)

A. Marcia Barbosa

## See Also

`substr`, `strsplit`

### Examples

```
data(rotifers)

head(rotifers)

## add a column to 'rotifers' with shorter versions of the species names:

## Not run:
rotifers$spcode <- spCodes(rotifers$species, sep.species = "_", nchar.gen = 1,
nchar.sp = 4, nchar.ssp = 0, sep.spcode = ".")

# this produces an error due to resulting species codes not being unique

## End(Not run)

rotifers$spcode <- spCodes(rotifers$species, sep.species = "_", nchar.gen = 1,
nchar.sp = 5, nchar.ssp = 0, sep.spcode = ".")

# with a larger number of characters from the specific name,
# resulting codes are now unique

## check out the result:
head(rotifers)
```

---

## splist2presabs — *Convert a species list to a presence-absence table*

---

### Description

This function takes a locality+species dataset in long (stacked) format, i.e., a matrix or data frame containing localities in one column and their recorded species in another column, and converts them to a presence-absence table (wide format) suitable for mapping and for computing distributional similarities (see e.g. `simMat`). Try out the Examples below for an illustration).

### Usage

```
splist2presabs(data, sites.col, sp.col, keep.n = FALSE)
```

### Arguments

| | |
|---|---|
| `data` | a matrix or data frame with localities in one column and species in another column. Type `data(rotifers); head(rotifers)` for an example. |
| `sites.col` | the name or index number of the column containing the localities in `data`. |
| `sp.col` | the name or index number of the column containing the species names or codes in `data`. |
| `keep.n` | logical value indicating whether to get in the resulting table the number of times each species appears in each locality; if `FALSE` (the default), only presence (1) or absence (0) is recorded. |

### Value

A data frame containing the localities in the first column and then one column per species indicating their presence (or their number of records if `keep.n = TRUE`) and absence. Type `data(rotif.env); head(rotif.env[,18:47])` for an example.

### Author(s)

A. Marcia Barbosa

### See Also

`table`

### Examples

```
data(rotifers)

head(rotifers)

rotifers.presabs <- splist2presabs(rotifers, sites.col = "TDWG4",
sp.col = "species", keep.n = FALSE)

head(rotifers.presabs)
```

---

## stepByStep *Analyse and compare stepwise model predictions*

---

### Description

This function builds a generalized linear model with forward stepwise inclusion of variables, using AIC as the selection criterion, and provides the values predicted at each step, as well as their correlation with the final model predictions.

### Usage

```
stepByStep(data, sp.col, var.cols, family = binomial(link = "logit"),
Favourability = FALSE, trace = 0, cor.method = "pearson")
```

### Arguments

| | |
|---|---|
| `data` | a data frame containing your target and predictor variables. |
| `sp.col` | index number of the column of `data` that contains the target variable. |
| `var.cols` | index numbers of the columns of `data` that contain the predictor variables. |
| `family` | argument to be passed to the `glm` function indicating the family (and error distribution) to use in modelling. The default is binomial distribution with logit link (for binary target variables). |

| | |
|---|---|
| `Favourability` | logical, whether to apply the Favourability function to remove the effect of prevalence from predicted probability (Real et al. 2006). Applicable only to binomial GLMs. Defaults to FALSE. |
| `trace` | argument to pass to the `step` function. If positive, information is printed during the stepwise procedure. Larger values may give more detailed information. The default is 0 (silent). |
| `cor.method` | character string to pass to function `cor` indicating which coefficient should be used for correlating predictions at each step with those of the final model. Can be "pearson" (the default), "kendall", or "spearman". |

## Details

Stepwise variable selection often includes more variables than would a model selected after examining all possible combinations of the variables (e.g. with packages **MuMIn** and **glmulti**). The `stepByStep` function can be useful to assess if a stepwise model with just the first few variables could already provide predictions very close to the final ones (see e.g. Fig. 3 in Munoz et al., 2005). It can also be useful to see which variables determine the more general trends in the model predictions, and which just add (local) additional nuances.

## Value

This function returns a list of the following components:

| | |
|---|---|
| `predictions` | a data frame with the model's fitted values at each step of the variable selection. |
| `correlations` | a numeric vector of the correlation between the predictions at each step and those of the final model. |
| `variables` | a character vector of the variables in the final model, named with the step at which each was included. |
| `model` | the resulting model object. |

## Author(s)

A. Marcia Barbosa

## References

Munoz, A.R., Real R., BARBOSA A.M. & Vargas J.M. (2005) Modelling the distribution of Bonelli's Eagle in Spain: Implications for conservation planning. Diversity and Distributions 11: 477-486

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

## See Also

`step`, `glm`, `modelTrim`

### Examples

```
data(rotif.env)

stepByStep(data = rotif.env, sp.col = 18, var.cols = 5:17,
cor.method = "spearman")

stepByStep(data = rotif.env, sp.col = 18, var.cols = 5:17,
cor.method = "spearman", Favourability = TRUE)

stepByStep(data = rotif.env, sp.col = 9, var.cols = c(5:8, 10:17),
family = poisson)
```

---

## timer *Timer*

---

### Description

Reporting of time elapsed since a given start time. This function is used internally by other functions in the package.

### Usage

```
timer(start.time)
```

### Arguments

| | |
|---|---|
| `start.time` | A date-time object of class `POSIXct`, e.g. as given by `Sys.time`. |

### Value

The function returns a message informing of the time elapsed since the input `start.time`.

### Author(s)

A. Marcia Barbosa

### See Also

`Sys.time`, `proc.time`, `difftime`

### Examples

```
# get starting time:
start <- Sys.time()

# do some random analysis:
sapply(rnorm(50000), function(x) x*5)

# see how long it took:
timer(start)
```

transpose *Transpose (part of) a matrix or dataframe*

## Description

This function transposes (a specified part of) a matrix or data frame, optionally using one of its columns as column names for the transposed result. It can be useful for turning a species presence-absence table into a regional species composition table.

## Usage

```
transpose(data, sp.cols = 1:ncol(data), reg.names = NULL)
```

## Arguments

| | |
|---|---|
| data | a matrix or data frame containing the species occurrence data to transpose. |
| sp.cols | names or index numbers of the columns containing the species occurrences in data which are meant to be transposed. |
| reg.names | name or index number of the column in data containing the region names, to be used as column names in the transposed result. |

## Value

This function returns the transposed sp.cols of data, with the column specified in reg.names as column names.

## Author(s)

A. Marcia Barbosa

## See Also

t

## Examples

```
data(rotif.env)

head(rotif.env)

names(rotif.env)

rotif.reg <- transpose(rotif.env, sp.cols = 18:47, reg.names = 1)

head(rotif.reg)
```

triMatInd *Triangular matrix indices*

## Description

This function outputs the indices of one triangle (the lower one by default) of an input square matrix. It is used by `simMat` and, for large matrices, makes it faster than e.g. with `lower.tri` or `upper.tri`.

## Usage

```
triMatInd(mat, lower = TRUE, list = FALSE)
```

## Arguments

| | |
|---|---|
| `mat` | a square matrix. |
| `lower` | logical indicating whether the indices should correspond to the lower triangle. The default is TRUE; FALSE produces the upper triangle indices. |
| `list` | logical indicating whether the results should be output as a list instead of a matrix. The default is FALSE. |

## Value

The indices (row, column) of the elements of the matrix that belong to the requested triangle.

## Author(s)

A. Marcia Barbosa

## References

http://stackoverflow.com/questions/20898684/how-to-efficiently-generate-lower-triangle-indices-of-a-symmetric-matrix

## See Also

`lower.tri`, `upper.tri`

## Examples

```
mat <- matrix(nrow = 4, ncol = 4)
mat
triMatInd(mat)
triMatInd(mat, list = TRUE)
```

# Index