Package 'simba'

August 15, 2010

Type Package

Version 0.3-2

Date 2010-08-14

Title A Collection of functions for similarity analysis of vegetation data

Author Gerald Jurasinski <gerald.jurasinski@uni-rostock.de>, with contributions from Vroni Retzer <vroni.retzer@gmx.de>

Maintainer Gerald Jurasinski <gerald.jurasinski@uni-rostock.de>

Depends R (>= 2.9.0), stats, vegan

Suggests gplots, geoR, plotrix

Description Besides functions for the calculation of similarity and multiple plot similarity measures with binary data (for instance presence/absence species data) the package contains some simple wrapper functions for reshaping species lists into matrices and vice versa and some other functions for further processing of similarity data (Mantel-like permutation procedures) as well as some other useful stuff for vegetation analysis.

License GPL (>= 2)

URL http://www.r-project.org, http://www2.auf.uni-rostock.de/loe/ma/simba.html

LazyLoad yes

R topics documented:

simba-package . . . 2
abis . . . 3
ads.ternaries . . . 4
aslopect . . . 6
auc . . . 7
bb2num . . . 8
bcoov . . . 9
bernina . . . 11
boxes . . . 12
com.sim . . . 13
dfcor . . . 15

diffmean . . . 17
diffmich . . . 18
diffslope . . . 20
direct . . . 23
dist.tmp . . . 24
hexgrid . . . 25
liste . . . 26
makead . . . 28
mama . . . 30
occ.time . . . 31
pcol . . . 34
plot.mrpp . . . 37
rin . . . 38
sim . . . 44
sim.tmp . . . 51
sim.yo . . . 53
simbadocs . . . 55
symbol.size . . . 56

Index **58**

`simba-package` *Calculate similarity measures for binary data*

Description

Besides a function for the calculation of similarity measures with binary data (for instance presence/absence species data) the package contains some simple wrapper functions for reshaping species lists into matrices and vice versa and some other functions for further processing of similarity data.

Details

Package:	simba
Type:	Package
Version:	0.3-1
Date:	2010-07-25
License:	GPL version 2

The functions in this package can be used to calculate similarities between species records (in binary format). Functions related to the correlation of similarity matrices and some other useful functions for the analysis of spatial patterns and their change in time are included as well.

Author(s)

Gerald Jurasinski `<gerald.jurasinski@uni-rostock.de>`, with some contributions by Vroni Retzer `<vroni.retzer@gmx.de>`

References

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

Wilson, M. V. & Shmida, A. (1984) Measuring beta-diversity with presence-absence data. – Journal of Ecology 72, 1055–1064.

Gower, J.C. and Legendre, P. (1986) Metric and Euclidean properties of dissimilarity coefficients. *Journal of Classification*, **3**, 5–48.

Faith, D. P, Minchin, P. R. and Belbin, L. (1987). Compositional dissimilarity as a robust measure of ecological distance. *Vegetatio* 69, 57–68.

Krebs, C. J. (1999). *Ecological Methodology.* Addison Wesley Longman.

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

Mountford, M. D. (1962). An index of similarity and its application to classification problems. In: P.W.Murphy (ed.), *Progress in Soil Zoology*, 43–50. Butterworths.

Wolda, H. (1981). Similarity indices, sample size and diversity. *Oecologia* 50, 296–302.

See Also

`vegdist`, `dist.binary`, `dsvdis`, `dist`, `reshape`, `cor.test`

Examples

```
##see functions for examples
```

`abis` *Vegetation and environmental data recorded in a Tundra ecosystem in Northern Sweden (Southeast of Abisko).*

Description

Vegetation and related data from field sites in a Tundra Ecosystem in Northern Sweden. `abis.spec` holds abundance information of 158 species on 61 sites.

Usage

```
data(abis)
```

Format

- abis.env contains some environmental variables from the Abisko field sites. These are:
  - `X` Numeric: The x-coordinates of the field sites (utm). The sites are arranged in a regular equidistant grid. Each site is a hexagon plot with 5m radius.
  - Y Numeric: The y-coordinates of the geographic position of the field sites.
  - alt Integer: Height a.s.l. of the field sites.
  - aspect Exposition of the sites. In degree from 0 to 180/-180. Counts negative in westward direction and positive in eastward direction.
  - slope Numeric: the inclination of the sites in degree
  - pH Numeric: pH value of soil samples taken on the sites (mixed samples from three
  - n.spec Integer: Number of species on the site.
  - shannon Numeric: Shannon index calculated from the species data.
  - even Numeric: Evenness calculated from the species data.
  - simps Numeric: Simpson diversity index for the site.

- ephwtrs1 Numeric: Cover of ephemeral waters
- ephwtrs2 Numeric: Cover of dried out ephemeral waters?
- roughn Numeric: Roughness index for the site. Obtained through calculating the ratio between the length of the straight lines dissecting the hexagonal plot and the lengths of lines along this dissections but exactly following the surface.
- stones.cov Numeric: Proportion of area of site covered by rocks which are covered with vegetation.
- stones.uncov Numeric: Proportion of area of site with uncovered, bare rocks.
- bare.soil Numeric: Open soil (most often sandy substrates).
- water Numeric: Proportion of area covered by permanent water bodies.
- bog Numeric: Proportion of area covered with boggy depressions.
- lemming Numeric: Counts of Lemming feces.
- elk Numeric: Counts of Elk feces.
- ren Numeric: Counts of Ren feces.
- ripa Numeric: Counts of Ripa feces.
- fox Numeric: Counts of Fox feces.
- grubbings Numeric: Amount of mole grubbings.
- tread Numeric: Amount of hoof tread.
- betula Numeric: Cover of *Betula*.
- CEC Numeric: Cation Exchange Capacity measured from the mixed soil sample taken in the field.
- base.satur Numeric: Base saturation.
- heath Binary: Does the site belong to the vegetation type 'heath'?
- shrubs Binary: Does the site belong to the vegetation type 'shrubby vegetation'?
- protect Binary: Does the site belong to the vegetation type 'protected snow heath'?

Source

Jurasinski G, Jentsch A, Retzer V, Beierkuhnlein C (submitted) Assessing gradients in species composition with multiple plot similarity coefficients. *Ecography*

Rettenmaier, N. 2004. Räumliche Muster der Biodiversität in der skandinavischen Tundra - Diploma thesis (unpublished), Department of Biogeography, University of Bayreuth, pp. 96.

Examples

```
data(abis)
```

`ads.ternaries` *Artificial data-set for studying the mathematical behavior of asymmetric similarity coefficients*

Description

Artificial data-set as utilized in Koleff et al. 2003, and Jurasinski 2007 to study the mathematical behavior of asymmetrical similarity indices. The corresponding values of all indices computable with `sim` are contained. See example!

Usage

```
data(ads.ternaries)
```

Format

- ads.ternaries`data.frame` with the three matching components of asymmetric binary similarity measures `(a, b, c)` with all possible combinations of these components derived from a virtual data-set with 100 variables (species). These are the first three columns. The preceding columns contain the values of the similarity coefficients computable with `sim` according to the three matching components. This information can be used to study the mathematical behavior of the indices. See example

Source

Jurasinski, G. (2007) Spatio-Temporal Patterns of Biodiversity and their Drivers - Method Development and a Case Study from Northeastern Morocco. PhD-Thesis, Department of Biogeography, University of Bayreuth

Koleff, P., Gaston, K. J. & Lennon, J. J. (2003) Measuring beta diversity for presence-absence data. *Journal of Animal Ecology* **72**: 367-382.

Examples

```
data(ads.ternaries)
library(plotrix)
##take any index you want to study, see the help for sim() for available
##asymmetric indices or the names of the data.frame:
names(ads.ternaries)

##make a tmp from the index you want to study (we perform a transformation
##to obtain values between 0 and 1). if you want another index,
##just change the name in the next line:
tmp <- ads.ternaries$mountford
tmp <- (tmp-min(tmp))/max(tmp)
triax.plot(ads.ternaries[,c(2,3,1)], main="mountford",
col.symbols=grey(seq(0.1,1,0.1))[floor((tmp*100)/5)+1], pch=16)

##don't wonder: mountford is strange, just try another one:
##this time with rainbow-colors
tmp <- ads.ternaries$soerensen
tmp <- (tmp-min(tmp))/max(tmp)
triax.plot(ads.ternaries[,c(2,3,1)], main="sørensen",
col.symbols=rainbow(10)[floor((tmp*100)/10)+1], pch=16)

##and an interesting shape: routledge in greyscale...
tmp <- ads.ternaries$routledge
tmp <- (tmp-min(tmp))/max(tmp)
triax.plot(ads.ternaries[,c(2,3,1)], main="routledge",
col.symbols=grey(seq(0.1,1,0.1))[floor((tmp*100)/5)+1], pch=16)
```

`aslopect` *Calculate similarity of plots based on slope aspect and inclination*

Description

Allows for the comparison of plots regarding the two variables slope aspect and slope inclination at once. To obtain a distance measure integrating aspect and inclination the model of a unit sphere is used and great-circle distances between virtual locations are calculated. For each plot a virtual location on the sphere is defined using the values for aspect as longitude and 90°-inclination as latitude. See `details` for more...

Usage

```
aslopect(asp, slo, names=rownames(asp), fc=FALSE, listout = FALSE)
```

Arguments

`asp`	Numeric vector with aspect values, given in degree. Expects values between 0° (North) and 180°/-180° (South). Eastward directions count positive, westward directions count negative. Alternatively the aspect values can be given in degrees of a full circle up to 359° starting from 0° (North). In the latter case fc has to be set to `TRUE`.
`slo`	Numeric vector with slope inclination values. Expects values between 0°(flat) and 90° (vertical wall).
`names`	Plot names, defaults to the rownames of `asp`, but a separate vector can be specified. Its length has to match the length of `asp` and `slo`.
`fc`	Triggers conversion from full circle degrees to half circle degrees. Set this to `TRUE` if full circle degrees (from 0° to 360°) are given in `asp`.
`listout`	Shall the results be given in list-format (`data.frame`). Defaults to FALSE which means that a matrix of class `dist` is returned.

Details

To obtain a distance measure integrating aspect and inclination the model of a unit sphere is used and great-circle distances between virtual locations are calculated. For each plot a virtual location on the sphere is defined using the values for aspect as longitude and 90°-inclination as latitude. This means that as long as the inclination is low the virtual points are located in the pole region so that, regardless of aspect, plots with low inclination are rather close to each other regarding these qualities. The idea behind is, that solar radiation, wind or other factors highly depending on aspect and inclination are not really differing between plots with different aspect as long as the slope is low. The longitude values on the unit sphere are derived from the values of slope aspect. The equator of the sphere is thought as the compass circle. The Prime Meridian of the virtual sphere is the great circle through North and South of the compass. As in geographic terms longitude counts positive in Eastern and negative in Western direction. With ϕ = latitude = 90°-inclination and λ = longitude = aspect the great-circle distance between A and B can be calculated with the following formula.

$$sim = \zeta = \arccos\Big(\sin(\phi_A)\cdot\sin(\phi_B) + \cos(\phi_A)\cdot\cos(\phi_B)\cdot\cos(\lambda_B - \lambda_A)\Big)$$

Value

Returns a `dist` object or a `data.frame` (depending if listout = FALSE or TRUE). As a unit sphere is used, the maximum distance between two inclination/aspect pairs is perimeter/2 of the sphere which is by definition π. To scale the possible distances between 0 and 1 the results are divided by π. Thus, a great-circle distance of 1 is rather scarce in the real world, however, two vertical rock walls with opposite aspect would share it. If listout = TRUE a `data.frame` with the following variables returns.

`NBX`	one of the compared sampling units
`NBY`	the other part of the pair
`x`	The returned aslopect value

Author(s)

Gerald Jurasinski

Examples

```
data(abis)
## identify columns with slope and aspect data
names(abis.env)

## calculate aslopect
abis.aslop <- aslopect(abis.env[,4], abis.env[,5])
```

`auc` *Calculate the area under a line(curve).*

Description

Calculates the **a**rea **u**nder a **c**urve (integral) by decomposing the curve (line) into rectangles and adding their area.

Usage

```
auc(x, y, below.zero = TRUE)
```

Arguments

`x`	Numerical vector giving the x cordinates of the points of the line (curve).
`y`	Numerical vector giving the x cordinates of the points of the line (curve). One can calculate the integral of a fitted line through giving a vector to `x` that spans `xlim` with small intervalls and predicting the y coordinates with `predict` and that `x`-vector as `newdata`. See example.
`below.zero`	How to handle y-data below zero? When below zero data represent proper data you'll want to substract the areas below the zero line from the area above the zero line to integrate the area under the curve. When below zero makes no sense for your question, you are able to set this to `FALSE`. Then, all 'negative' areas are just set to 0.

Value

Returns a numeric value that expresses the area under the curve. The unit depends from the input.

Author(s)

Gerald Jurasinski, <gerald.jurasinski@uni-rostock.de>

Examples

```
## Construct a data set (Imagine 2-hourly ghg emission data
## (methane) measured during a day).
## The emission vector (data in mg CH4 / m2*h) as a time series.
ghg <- ts(c(12.3, 14.7, 17.3, 13.2, 8.5, 7.7, 6.4, 3.2, 19.8,
22.3, 24.7, 15.6, 17.4), start=0, end=24, frequency=0.5)
## Have a look at the emission development.
plot(ghg)
## Calculate what has been emitted that day
## Assuming that emissions develop linearly between
## measurements
auc(time(ghg), ghg)

## The effect of below.zero:
## Shift data, so that we have negative emissions (immissions)
ghg <- ghg-10
## See the difference
plot(ghg)
abline(h=0)
## With below.zero=TRUE the negative emissions are substracted
## from the positive emissions
auc(time(ghg), ghg)
## With below.zero=TRUE the negative emissions are set to 0
## and only the emissions >= 0 are counted.
auc(time(ghg), ghg, below.zero=FALSE)
```

---

bb2num — *Transform Braun-Blanquet scale data to percentage cover values and vice versa*

Description

The function allows for an easy transformation of Braun-Blanquet (or any other phytosciological scale) data to numeric values for numerical analysis or vice versa. Be aware that this transformation is not losless regarding the information content.

Usage

```
bb2num(dat, from = c("r", "+", "1", "2", "3", "4", "5"),
to = c(0.1, 1, 5, 15, 37.5, 62.5, 87.5))
```

Arguments

dat	A species matrix.
from	Definition of the imput scale. Per default the transformation is done from the classic Braun-Blanquet scale to cover percentages. For transformation from numerical values to a phytosociological scale see details and example.
to	Definition of the output scale. The default are the recommended class centers when transforming BB to percentage cover values. Can be changed according to your needs. See details.

Details

When transforming from a phytosociological scale to Braun-Blanquet`from` and `to` have to be the same length. In case of transformation from numerical cover values to a phytosociological scale the function expects a numerical vector in `from` that, for each class, gives the lower and upper limit. Thus, `length(from) == 2*length(to)` in this case.

Value

Returns a species matrix in the specified format.

Author(s)

Gerald Jurasinski <gerald.jurasinski@uni-rostock.de>

Examples

```
## Create a species that occurs on 7 plots
## with all the different possibilities of
## the BB scale
spec <- c("r", "+", "0", "1", "2", "3", "4", "5")

## Create a highly artificial species matrix.
## All species are the same for simplicity
dat.bb <- data.frame(spec, spec, spec, spec, spec, spec)

## Transform from BB scale to percentage values
dat.proc <- bb2num(dat.bb)

## When transforming back the class definitions are a little
## more complicated. Just give the lower and upper limits for
## each class in the from vector class for class.
from <- c(0, 0.1, 0.1, 2, 2, 5, 5, 25, 25, 50, 50, 75, 75, 100)
to <- c("r", "+", "1", "2", "3", "4", "5")
bb2num(dat.proc, from=from, to=to)
```

---

bcoov *Calculate Bray-Curtis distance for only one variable*

Description

Calculates Bray-Curtis distance for only one variable: How dissimilar are the units regarding for instance pH?

Usage

```
bcoov(x, names, listout = FALSE)
```

Arguments

x	numeric vector with quantitative data from sampling-units, standardization may be applied before
names	where should the names be taken from, must be a vector of the same length as x and should give the names of the sampling-units
listout	Shall the result given in list-format (data.frame)? Defaults to FALSE

Details

It is just a simple way to calculate similarity based on only one variable. In the future there might be a possibility to choose from some functions. However, you could also use the difference in the data-values instead (which is the Euclidean distance in one dimensional space).

Value

Returns a dist object or a data.frame (in the case of listout = TRUE) with

NBX	one of the compared sampling units
NBY	the other part of the pair
x	The returned distance value. It is named like the tested variable

Author(s)

Gerald Jurasinski

See Also

vegdist, dist.quant, dsvdis

Examples

```
data(abis)
names(abis.env) ##take a look at the data
##calculate the similarity (Bray-Curtis) between the plots
##regarding pH
pH.dist <- bcoov(abis.env$pH, names=rownames(abis.env))

## directly give it as a list (data.frame)
pH.dist.ls <- bcoov(abis.env$pH, names=rownames(abis.env)
, listout=TRUE)
```

`bernina` *Repeated vegetation records from 7 Alpine summits of the Bernina range.*

Description

Repeated species records from 7 Alpine summits of the Bernina range. The data contains species data from repeated field recordings that were carried out on the summits between 1905 and 1907 (Ruebel 1912), in 1985 (Hofer 1992), and in 2003 (Walther et al. 2005). Data was compiled by Walther et al. (2005) and used in Jurasinski & Kreyling (2007) to investigate homogenization of Alpine summit floras.

Usage

```
data(bernina)
```

Format

- `veg` Complete pecies matrix that contains the species records from the repeated samplings. Integer entries of '0' and '1'. Hence, the matrix represents presence/absence information.

- `veg.lst` Complete pecies data in list format that contains the species records from the repeated samplings. Has three columns ....
  - `plot` Factor: The summit names. Repeat for each species and time period.
  - `spec` Factor: The species names.
  - `occurrence` Integer: The occurrence information. Because it is presence/absence it is '1' for all species in the list.

- `summits` Information on the repeatedly sampled summits of the Bernina range. Most information repeats. However, the year, and the number of species change from record year to record year.
  - `year` Numeric: The year of the recording.
  - `number` Numeric: The summit number. Can be used for efficient reference to the summits.
  - `summit` Factor: Names of the summits. Repeat per recording period.
  - `altitude` Numeric: Altitudes of the summits. Repeat per recording period.
  - `northing` Numeric: UTM coordinates of the summit. Repeat per recording period.
  - `easting` Numeric: UTM coordinates of the summit. Repeat per recording period.
  - `n.spec` Numeric: Number of species found on the summit. Changes with recording period.

- `years` Character: A vector of repeated character strings that give the information from which year the resepective parts of `veg.lst` are. Therefore `length(years) == nrow(veg.lst)`.

Source

Hofer HR, 1992. Veraenderungen in der Vegetation von 14 Gipfeln des Berninagebietes zwischen 1905 und 1985. *Ber. Geobot. Inst. Eidg. Tech. Hochsch. Stift. Ruebel Zuer.* 58: 39-54.

Jurasinski G & Kreyling J, 2007. Upward shift of alpine plants increases floristic similarity of mountain summits. *Journal of Vegetation Science* 18: 711–718.

Ruebel E, 1912. Pflanzengeographische Monographie des Bernina-Gebietes. *Engelmann, Leipzig, DE*.

Walther G-R, Beißner S, Burga CA, 2005. Trends in the upward shift of alpine plants. *Journal of Vegetation Science* 16: 541-548.

Examples

```
data(bernina)
```

`boxes` *An adaption of boxplot.n.*

Description

Uses `boxplot` to produce a boxplot, which is then annotated with the number of obbservations in each group. Does allow for more flexibility compared to `boxplot.n`. Default size of the text is bigger and per default the numbers are not plotted directly on the x-axis and their distance from the axis can be changed.

Usage

```
boxes(..., top = FALSE, shrink = 1, textcolor = NULL, yadj = NULL)
```

Arguments

`...`	Parameters passed to `boxplot`.
`top`	Should the numbers of observations be printed below or above the boxes? Defaults to below (top = FALSE).
`shrink`	Parameter to scale the size of the numbers of observations. Above 1 increases size, below 1 until 0 decreases size.
`textcolor`	Color of the text. Defaults to NULL which than uses the actual plotting colour of the graphics device.
`yadj`	Can be used to adjust the vertical plotting positions of the numbers of observations. Defaults to NULL - no adjustment.

Author(s)

Gerald Jurasinski

See Also

`boxplot`, `plot`, `boxplot.n`

Examples

```
data(abis)

## see environmental data (see documentation on data for details)
abis.env

## calculate the difference in similarities for the three major
## vegetation types
## therefore created a vector from the data expressing belonging
## to the vegetation types:
tcs.sub <- rep(0, 61)
tcs.sub[abis.env[,29]==1] <- 1
tcs.sub[abis.env[,30]==1] <- 2
tcs.sub[abis.env[,31]==1] <- 3

## calculate similarity (Jaccard) between all pairs of plots
abis.jacc <- sim(abis.spec, method="jaccard")

## make boxplots regarding the similarities for each vegetation
## type, including the number of pairs contained in each box.
boxes(as.matrix(abis.jacc)~tcs.sub, shrink=1.2)

## prettier
boxes(as.matrix(abis.jacc)~tcs.sub, notch=TRUE, col="grey50",
shrink=1.2, ylim=c(-0.1, 0.9))
```

---

com.sim *Compare mean similarity between subsets of data*

Description

Related to mrpp. Are the differences in mean similarity between data subsets significant? Function takes the whole data-set (species matrix) and a subsetting vector and computes a specified similarity between all sampling units (rows). Then subsets are compared regarding their mean similarity. Statistical inference is obtained through permutation.

Usage

```
com.sim(veg, subs, simil = "soerensen", binary = TRUE,
    permutations = 1000, alpha = 0.05, bonfc = TRUE, ...)
```

Arguments

veg	Species matrix with columns = sites, rows = species. Deliver presence/absence data or abundance data. However, binary has to be set accordingly.
subs	Vector containing the subset definition. Same entries are understood to indicate belonging to the same subset (can be characters, factors or numerics). For each subset similarities/distances are calculated. Then all subsets are compared regarding mean and variance of the similarities/distances.
simil	Sets the coefficient to be used for calculating similarities/distances. If binary = TRUE, see sim, otherwise see vegdist for possible choices.

binary	Changes the function used for the calculation of similarity/distance. If binary species data ist provided in veg keep the default (binary = TRUE). In this case sim is used to calculate the similarities. Set to FALSE when abundance or frequency data is provided. This calls vegdist to calculate the distances between sites in species similarity space.
permutations	Number of permutations performed to obtain the statistical inference. See Details.
alpha	Initial alpha level to test against. Defaults to 0.05.
bonfc	Shall Bonferroni correction be applied? Defaults to true.
...	Further arguments to functions.

Details

Entries of similarity/distance matrices are not independent. Therefore normal statistics might fail. One possibility is the application of permutation procedures. This means that the statistical distribution against which significance is tested is derived from the data.

Here it is implemented as follows: For each subset the similarities/distances between all sites (plots) are calculated with the specified coefficient. Then the resulting similarity/distance matrices are compared with diffmean. This is done for the comparison of each subset with each other subset. If specified (defaults to TRUE), Bonferroni correction is applied (to correct for multiple testing).

Depending on the number of subsets and the number of sites per subset it may take some seconds to be computed.

Value

Returns an object of class cslist containing the call to the function, the used method for similarity/distance calculation, a comparison matrix showing the connections between data-subsets (rows and columns connected with "*" are significantly different), the number of subsets involved, the number of permutations and a matrix giving information about the following components for each comparison between subsets:

X	Subset identifier for one of the compared subsets
Y	Subset identifier for the other compared subset
mean.x	Average distance/similarity for subset X.
mean.y	Average distance/similarity for subset Y.
diff	Difference in average distance/similarity for this comparison
sig	Significance of the difference in mean of the similarities.
sigs	Significance flag for the comparison ("*" means significant differences, "ns" means that the differences are not significant).
Fval	F-value for the Comparison.
sigF	Is F significant?
sigsF	Significance flag for F.

Author(s)

Gerald Jurasinski

See Also

mrpp for an anova like approach for comparing the differences of species data subsets.

Examples

```
data(abis)

## see environmental data (see documentation on data for details)
abis.env

## calculate the difference in similarities for the three major
## vegetation types
## therefore create a vector from the data expressing belonging
## to the vegetation types:
tcs.sub <- rep(0, 61)
tcs.sub[abis.env[,29]==1] <- 1
tcs.sub[abis.env[,30]==1] <- 2
tcs.sub[abis.env[,31]==1] <- 3

## calulate differences with Bray-Curtis as the distance measure
com.sim(abis.spec, tcs.sub, simil="bray", binary=FALSE)

## calculate differences with Soerensen as the similarity measure
com.sim(abis.spec, tcs.sub)
```

`dfcor` *Calculate permuted (Mantel) correlations between one and many variables*

Description

The function uses `permcor2` to calculate permuted correlation on vectors. One vector is compared to various vectors of the same length. Useful e.g. if one variable has to be tested against various variables.

Usage

```
dfcor(ox, y, method = "pearson", permutations = 1000, ...)
```

Arguments

`ox`	Numeric vector. If it is a similarity matrix (i.e. a `dist` object, extract vector via `as.vector(x)` beforehand).
`y`	A `data.frame` containing numeric vectors to correlate `ox` with. Number of rows has to equal the length of `x`
`method`	Method for correlation. Defaults to "pearson". See `cor` for other possibilities.
`permutations`	Number of permutations. Defaults to 1000, which gives reasonable results and allows to test against alpha = 0.001.
`...`	Further arguments passed to internal functions (i.e. to `cor`).

Details

`dfcor` is a wrapper for permcor2, which is usually called as a part of `pcol`. Here, the numeric vector in `x` is compared to each column vector of `y`.

Value

A list with the following:

call	The function call
method	P-value obtained by testing the initial correlation against the permuted correlation values.
out	A table with statistics. See details below.
gesN	The included number of cases.
strata	The number of variables against which x was tested.
permutations	The number of permutations.
	The included printing method gives nice output (where information appears in a slightly different order) ending with the table of out. It is based on a data.frame with nrow = ncol(y) giving the statistics for the correlation between x and each column of y as follows.
corr	Correlation value (regarding to method).
sig	P-value obtained by testing the initial correlation against the permuted correlation values.
nop	Number of included pairs. The function tests for complete cases before calculation starts. Pairs containing NA's are not included.
miss	Number of missing pairs.

Note

Maybe pcol will get this functionality in future releases.

Author(s)

Gerald Jurasinski

References

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

See Also

For related functions of simba permcor, permcor2, mancor, pcol. Further see mantel of package vegan for a different implementation of permuted correlation on distance matrices.

Examples

```
## load included data
data(abis)

## obtain several similarity matrices and correlate them
## with one other matrix.
# calculate soerensen similarity for species composition data
abis.soer <- sim(abis.spec, listout = TRUE)[,3]
# calculate euclidean distances between the plot locations
abis.geodist <- dist(abis.env[,1:2])
```

```
# calculate the ecological distance according to slope and aspect
abis.aslopect <- with(abis.env, aslopect(aspect, slope))
# calculate the ecological distance according to disturbance
abis.pert <- dist(abis.env[,19-25])

# put distances according to environment together
abis.envdist <- data.frame(dist = c(abis.geodist),
aslopect = c(abis.aslopect), pert = c(abis.pert))

# no mantel correlation between those constructed
# variables and the similarity in species composition
dfcor(abis.soer, abis.envdist)
```

`diffmean` *Calculate the difference in Mean between two vectors*

Description

The function can be used to calculate the difference in mean between two vectors. Statistical inference is obtained through permutation. F-ratio is also calculated. For data which is not normally distributed or lacks independence. The plotting method plots the actual values of the difference in mean and F against an histogram of the results of the permuted runs.

Usage

```
diffmean(x, y, permutations = 1000)
## S3 method for class 'dmn':
plot(x, y, which=3, two=2, ...)
```

Arguments

`x`	Numeric vector. For the plotting method the `dmn`-object which should be printed (results from a `diffmean` operation). Plotting object in the plotting method.
`y`	Numeric vector. Plotting object in the plotting method, optional when `x` has appropriate structure
`permutations`	Number of permutations.
`which`	which histogram should be plotted? 1 triggers the histogram for difference in mean, 2 the one for F. It defaults to 3: both histograms are plotted. If it is changed from default, the next argument (`two`) is automatically set to 1!
`two`	Should the histograms be printed on a divided display? And how? Can only be set if `which` is set to 3. Defaults to 2, which means that the display is divided in two halfs and the histogram-plots are plotted side by side. 3 causes histograms to be plotted one on top of the other. If `two` = 1, the display is NOT automatically divided. Might be useful if more than one `dmn`-objekt is to be plotted on one display. Otherwise the function overrides the actual display settings.
`...`	Further arguments to the plotting method.

Details

The two vectors do not need to share the same length but they should not be too different. Otherwise the function might give spurious results.

Value

Returns a list giving the function call, the difference in Mean, the mean of vector x and y, the mean of means, the F-value, the significance of the difference in Mean and the significance of F, as well as the number of permutations. The results of the permutation runs can be retrieved with `result$bootsM` (for the difference in mean) and `result$bootF` (for the F-values). There is a plot method for easily illustrating the test. The difference is plotted against an histogram displaying the distribution of the permuted values.

Author(s)

Gerald Jurasinski

See Also

`diffslope`, `diffmich`

Examples

```
data(abis)

## create subsetting vector describing the belonging to different
## vegetationtypes
tcs.sub <- rep(0, 61)
tcs.sub[abis.env[,29]==1] <- 1
tcs.sub[abis.env[,30]==1] <- 2
tcs.sub[abis.env[,31]==1] <- 3

## check distribution
summary(as.factor(tcs.sub))

## compare vegetation types "shrubby vegetation" (shrub=2) and
## "protected by snowcover" (protect=3) regarding difference in
## similarities
abis2.soer <- sim(abis.spec[tcs.sub==2,])
abis3.soer <- sim(abis.spec[tcs.sub==3,])
abis.23cmp <- diffmean(abis2.soer, abis3.soer)
```

---

diffmich *Calculate the difference in parameters of a Michaelis-Menten kinetik fitted to (PAM) data*

Description

The function can be used to calculate the difference in the two parameters of the Michaelis-Menten Kinetik $y = \frac{a*x}{b+x}$ between two datasets containg each two vectors. Through permutation it is possible to compute significance of the difference. `fitmich` is used to calculate the Michaelis-Menten fit to the data. With the corresponding plot method a plot of the actual difference in the parameters against a histogram of the permuted values can easily be achieved.

Usage

```
diffmich(x1, y1, x2, y2, permutations = 1000, a=3, b=0.5,
trace=FALSE, ...)

fitmich(x, y, a=3, b=0.5)

## S3 method for class 'diffmich':
plot(x, y, which=3, two=2, ...)
```

Arguments

x1	Vector containing an independent variable, for instance PAR measurements.
y1	Vector containing a variable dependent on x1 (for instance ETR measurements). Must have the same length as x1.
x2	Vector containing a second independent variable (for instance PAR measurements).
y2	Vector containing a variable dependent on x2 (for instance ETR measurements). Must have the same length as x2.
permutations	Number of permutations.
a	start value for parameter a, defaults to 3, usually there is no change necessary, but if the function gets trapped in the first run, changing the parameters might solve the problem.
b	Start value for parameter b, defaults to 0.5.
trace	set to TRUE for displaying the progress of the calculation
...	Arguments to other functions (for instance to lm, which is used to calculate the regression lines)
x	Vector containing an independent variable, for instance PAR measurements. Function fitmich is usually called only internally by diffmich. Plotting object in the plot method.
y	Vector containing a variable dependent on x1 (for instance ETR measurements). Must have the same length as x1. Plotting object in the plotting method, optional when x has appropriate structure
which	Which histogram should be plotted? 1 triggers the histogram for parameter a, 2 the one for parameter b. It defaults to 3: both histograms are plotted. If it is changed from default the next argument (two) is automatically set to 1!
two	Should the histograms be printed on a divided display? And how? Can only be set if which is set to 3. Defaults to 2, which means that the display is divided in two halfs and the histogram-plots are plotted side by side. 3 causes histograms to be plotted one on top of the other. If two = 1, the display is NOT automatically divided. Might be useful if more than one diffmich-objekt is to be plotted on one display. Otherwise the function overrides the actual display settings.

Details

As the function was initially built to easily calculate the difference of parameters of the Michaelis-Menten Kinetik for PAM measurements, the independent vectors are meant to contain PAR values whereas the dependent vectors should represent ETR values. But you can use it for anything else which can be fitted with Michaelis-Menten. The vectors belonging together are formed into a data.frame. For each permutation run the rows are interchanged randomly between the two

data.frames and the difference in the parameters is calculated and collected into a vector. The p-value is then computed as the ratio between the number of cases where the differences in Parameter exceed the difference in parameter of the inital configuration and the number of permutations.

As it uses a for loop it takes a while to calculate. So get a coffee while it is running, or set trace = TRUE to avoid boring moments ...

Value

Returns a diffmich-object with the function call, the difference in the two parameters and their significance. Furthermore the number of permutations. If you want to change the way fitmich is computed you can change the starting values. Per default it is calculated with starting values a=3 and b=0.5. There's no change needed unless the function gets trapped.

Author(s)

Gerald Jurasinski

See Also

nls, sample

---

diffslope *Calculate the difference in slope or intercept of two regression lines*

Description

The function can be used to calculate the difference in slope between two datasets containing each two vectors. Follows an idea of Nekola & White (1999) for calculating the statistical inference of the difference in slope between two regression lines. diffslope2 has the same purpose as diffslope but implementation is without for-loop. The plot method allows easy plotting of the actual difference in slope against the distribution of permuted values.

Usage

```
diffslope(x1, y1, x2, y2, permutations = 1000, ic = FALSE,
resc.x = FALSE, resc.y = TRUE, trace=FALSE, ...)

diffslope2(x1, y1, x2, y2, permutations = 1000, resc.x = FALSE,
        resc.y = TRUE, ...)

diffic(x1, y1, x2, y2, permutations = 1000, resc.x = FALSE,
        resc.y = FALSE, trace=FALSE, ...)

## S3 method for class 'dsl':
plot(x, y, ...)
```

Arguments

x1	vector containing an independent variable (for instance distance between plots).
y1	vector containing a variable dependent on x1 (for instance similarity between the plots. must have the same length as x1.
x2	vector containing a second independent variable (for instance distance between plots). can be the same as in x1.
y2	vector containing a variable dependent on x2 (for instance similarity between the plots. must have the same length as x2.
permutations	number of permutations
ic	Shall the difference in intercept be tested? Defaults to FALSE.
resc.x	Shall the values of the independent variables be rescaled to a common mean? Defaults to FALSE.
resc.y	Shall the values of the dependent variables be rescaled to a common mean? Defaults to TRUE (Nekola & White 1999) for diffslope and to FALSE for diffic.
trace	Set to true if progress shall be printed with increasing numbers. Defaults to FALSE
...	Arguments to other functions (for instance to lm, which is used to calculate the regression lines).
x	dsl-object (given back by diffslope or diffic) which is to be plotted.
y	Plotting object, usually not necessary

Details

diffslope: As the function was initially build to easily calculate the difference in slope between the regression lines of distance decay plots, the independent vectors are meant to contain distance values whereas the dependent vectors should represent similarity values. But you can use it for anything else, as you wish. The vectors belonging together are formed into a data.frame. For each permutation run the rows are interchanged randomly between the two data.frames and the difference in slope calculated thereafter is calculated and collected into a vector. The p-value is then computed as the ratio between the number of cases where the differences in slope exceed the difference in slope of the inital configuration and the number of permutations.

The same applies to diffic. However, this function tests whether the intercepts of the two relationships are significantly different. Although resc.y has been kept as an option, it is not wise to do so, when one is testing for differences in intercept.

If the difference in slope returns negative, the slope (distance decay) of the second relationship is less pronounced, if it returns positive, the second relationship exhibits a stronger distance decay (slope) than the first. This holds for distance decay relationships. If y increases with x, it is vice versa.

As it uses a for loop, it takes a while to calculate. So get a coffee while it is running, or set trace to TRUE to avoid being bored ...

Value

Returns a list giving the function call, the difference in slope, the significance of this difference, and the number of permutations. If you want to change the way lm is computed you must send the arguments to lm via .... Per default it is calculated with the default arguments of lm.

In case of diffic there is still differences in slope reported (although differences in intercept have been calculated). So it's just a false label here. This will be updated soon.

Author(s)

Gerald Jurasinski

References

Nekola, J. C. and White, P. S. (1999) The distance decay of similarity in biogeography and ecology. Journal of Biogeography 26: 867-878.

Steinitz, O., Heller, J., Tsoar, A., Rotem, D. and Kadmon, R. (2005) Predicting Regional Patterns of Similarity in Species Composition for Conservation Planning. Conservation Biology 19: 1978-1988.

Steinitz, O., Heller, J., Tsoar, A., Rotem, D. and Kadmon, R. (2006) Environment, dispersal and patterns of species similarity. Journal of Biogeography 33: 1044-1054.

See Also

lm, sample

Examples

```
data(abis)
names(abis.env) ##take a look at the data
pert.dist <- 1-vegdist(abis.env[,19:25], "euclidean")
##calculate the distance (Euclidean) between the plots
##regarding disturbance variables

soil.dist <- 1-vegdist(abis.env[,c(6,27:28)])
##calculate the similarity (Bray-Curtis) between the plots
##regarding soil parameters

##calculate geographical distance between plots
coord.dist <- dist(abis.env[,1:2])

##transform all distance matrices into list format:
struc.dist.ls <- liste(pert.dist, entry="BC.struc")
soil.dist.ls <- liste(soil.dist, entry="BC.soil")
coord.dist.ls <- liste(coord.dist, entry="dist")

##create a data.frame containg plot information, geographical
##distance,similarity of soil parameters, and similarity of
##structural parameters:

df <- data.frame(coord.dist.ls, soil.dist.ls[,3], struc.dist.ls[,3])
names(df) ##see names

##give better names:
names(df)[4:5] <- c("soil","struc")
attach(df)

##prepare graphics device:
par(mfrow=c(2,1))

##plot and compare distance decay (decrease of similarity with
##distance):
plot(dist, soil)
plot(dist, struc)
```

```
##remove problematic zero entries:
df <- subset(df, struc != 0)

##plot again, this time with regression lines (in red for better
##visability):
detach(df)
attach(df)
plot(dist, soil)
abline(lm(soil~dist), col="red4")
plot(dist, struc)
abline(lm(struc~dist), col="red4")
##is the slope significantly different?
res <- diffslope(dist, soil, dist, struc)
res2 <- diffic(dist, soil, dist, struc)

##go for a coffee, as it takes a while...
```

`direct` *Obtain Direction Classes from Geographic Coordinates*

Description

The functions calculates direction classes from geographic coordinates (not lat/lon). All possible connections between these points are established and the direction of each link is calculated. This is followed by a designation of direction-classes.

Usage

```
direct(coord, listout=FALSE)

direct2(coord, listout=FALSE)
```

Arguments

`coord`	A `data.frame` containing coordinates. Should have the same number of points as the data for which the direction-classes are calculated.
`listout`	Logical value, indicating whether the result is given back in (`data.frame`)-format instead of returning a `dist`-object.

Value

Returns a matrix containing the direction-classes of the connections between the coordinates as a `dist`-object. If `listout` = TRUE, the result is given as a list (`data.frame`).

`direct` returns 4 directions (North-South, Northwest-Southeast, West-East, Northeast-Southwest).

`direct2` returns 6 directions.

Note

as with `mantel` it takes a while to calculate

Author(s)

Gerald Jurasinski

See Also

mantel, cor.test

Examples

```
data(abis)
dirclass <- direct(abis.env[,1:2])
dirclass
dirclass.ls <- direct(abis.env[,1:2], listout=TRUE)
dirclass.ls
```

---

dist.tmp — *Calculate the distance between the instances or variables of two similar data-sets.*

Description

The function calculates the distance between the instances or variables of two data-sets, preferably two recordings from the same places/variables at different time steps. Could for example be useful to calculate the dissimilarity of species records from the same set of plots at different recording campaigns.

Usage

```
dist.tmp(x, y, method = "euclidean", margin = 1, adjust=TRUE)
```

Arguments

x	data.frame with numeric columns giving the data at time one
y	data.frame with numeric columns giving the data at time two
method	Which distance measure should be employed? The following measurees are available: "manhattan", "euclidean", "bray", "canberra", "kulczynski", "gower", "jaccard". See details.
margin	Shall the distance between the instances (rows, margin = 2), or the distance between the variables (columns, margin = 1) be calculated?
adjust	The default adjust = TRUE takes care of your input data. However, instances and variable names have to be unique, as the matching relies on these names. If your data meets this requirement the function automatically detects which instances and variables can be found in both data-sets and calculates the distance based on these data.

Details

The function provides just an alternative to code your repeated measurements from the same plots, calculate the distances between all instcances and sort out the distances between the same instances at different time steps. Here just the latter is calculated directly.

For details regarding the different distance coefficients, see vegdist. The formulae behind were taken from there. However, not all choices have been implemented here.

Value

Returns a vector of distances. Length depends on `margin`. If you choose `margin` = 1 it equals the number of instances (plots) in your two data-sets. If you choose `margin` = 2 it equals the number of variables (species) in your data-sets.

Author(s)

Gerald Jurasinski <gerald.jurasinski@uni-rostock.de>

References

See references in `vegdist`

See Also

`vegdist`, `dist`, `sim`

Examples

```
data(abis)

##just to see that the function takes care of matching instances
##and variables we compare abis.spec with a permuted version of
##itself and skip some instances and variables in the original matrix.
dist.tmp(abis.spec[5:50,3:120], apply(abis.spec, 2, sample), method="bray")
```

---

hexgrid *Produces the nodes of an equidistant grid.*

Description

Given the coordinates of a starting point (left upper point of the grid), the function produces the nodes of an equidistant grid. Extent and distance between plots can be specified.

Usage

```
hexgrid(x, y, r = 100, nro = 10, nco = 20)
```

Arguments

`x`	x-value of the starting point. Defaults to 0.
`y`	y-value of the starting point. Defaults to 0.
`r`	Distance between nodes. Defaults to 100.
`nro`	Number of rows in the grid. Defaults to 10
`nco`	Number of columns in the grid. Defaults to 20. They are doubled (see the grid) compared to the rows. That's why 2*`nro` produces a quadratic grid.

Details

If the overall shape of the grid is not square, the user has to delete by hand the superfluous units. Might get more flexible in future versions.

Value

Returns a data.frame giving informations on the produced point/unit/plot locations with the following columns:

ROW	Number of row in the grid to which the point/unit/plot belongs.
COL	Number of column in the grid to which the point/unit/plot belongs.
X	x-coordinate of the point.
Y	y-coordinate of the point.

Author(s)

Gerald Jurasinski

References

http://homepage.mac.com/terhorab/gerald/downloads/whyhexaagons.pdf

See Also

spsample

Examples

```
## produces a grid with r=400:
test.grd <- hexgrid(456000, 7356700, r=400)
## for plotting the following is recommended as it preserves
## real positions:
library(geoR)
points.geodata(coords=test.grd[,3:4], data=rnorm(nrow(test.grd)))
```

---

liste *Convert dist-objects and matrices to database format*

Description

Transposes dist objects to database list format (where each line represents a similarity value calculated between two plots, so the list has three columns containing information on plot x, plot y and information on similarity/dissimilarity). This might be useful if further database processing is intended. If the given matrix is not of class dist, the whole matrix is converted. This might be handy if species matrices are to be converted to database format.

Usage

```
liste(obj, x="NBX", y="NBY", entry=NULL, factorize=TRUE,
splist=FALSE)
```

Arguments

obj	A distance object as it is returned from sim, dist, vegdist or dist.binary or a similar matrix with class="dist". If the matrix is not of class dist a data.frame is expected and all entries are converted. Species matrices are assumed to contain sites (or plots) in rows and species in columns.
x	How the second column should be named, standard is that it contains "NBX" the neighbour X. Obsolete when splist = TRUE.
y	How the second column should be named, standard is that it contains "NBY" the neighbour Y. Obsolete when splist = TRUE.
entry	How the third column should be named. If there is nothing given, it is named "we" (whatever). Obsolete when splist = TRUE.
factorize	If naming (first two) columns should be given as factors. Defaults to TRUE.
splist	Set to TRUE if a species matrix is transformed to a database list format. Automatically removes entries with zero occurrence of the species' and names columns correspondingly (see Value).

Details

Column x represents the column names of the input matrix. So if you want to reshape a species matrix, x will be the species names and y will be the plot names. If it is needed vice versa, you have to change column order. For convenience you can set splist = TRUE and you will get a species list in database format with columns representing the plot, species and occurrence information. Furthermore zero occurrences are already omitted.

Value

Returns a data.frame with three columns:

Col1	The column names of the input matrix. If it is a dist-object only the lower triangle is used. Named plot when a species matrix is transformed with splist = TRUE.
Col2	The row names of the input matrix. Named spec when a species matrix is transformed with splist = TRUE.
Col3	The respective matrix entries. Named occ when a species matrix is transformed with splist = TRUE.

Author(s)

Gerald Jurasinski

See Also

reshape, data.frame. It resembles functionality of reshape - which is more flexible (but also more complex).

Examples

```
data(abis)
## there are empty species entries:
sum(colSums(abis.spec)==0)
```

```
## remove empty species
abis.spec <- abis.spec[,colSums(abis.spec)!=0]
abis.spec.ls <- liste(abis.spec, splist=TRUE)
```

`makead` *Create artificial data set (species matrix).*

Description

The functions allow for the automated creation of artificial data (species matrix). The user can choose between random organization or a gradient. The gradient can be defined via a gradient vector which allows for fine tuning of the gradient. `ads` has a different implementation and produces better results for gradients.

Usage

```
makead(nspec, nplots, avSR = NULL, anc = NULL, grad.v = NULL,
cf = 0.2, puq = 0.01)

ads(nspec, nplots, avSR = NULL, anc = NULL, grad.v = NULL,
reord = TRUE, cf = 0.2, puq = 0.01)

ads.hot(nspec, nplots, avSR = NULL, anc = NULL, grad.v = NULL,
frac=0.5, reord=TRUE, cf=0.2, puq=0.01)

ads.fbg(nspec, nplots, grad.v, n.iter = 100, method = "ads", ...)
```

Arguments

`nspec`	Numbers of species you want to be in the data-set. Meaningless if `anc` != NULL.
`nplots`	Numbers of plots you want to be in the data-set. Meaningless if `anc` != NULL.
`avSR`	Average species richness. If `anc` is given, it is calculated from the data when the default is not changed. If `avSR` != NULL, the given value is taken instead. In the actual version not implemented in `ads`.
`anc`	If a model species matrix is available (either a real data-set, or another artificial data-set) on which creation should be based, give it here. Rows must be plots and columns be species. The first three parameters are then obtained from this set. However average species richness (`avSR`) can still be given by the user.
`grad.v`	A numeric vector describing the gradient, or - in case of `ads.hot` - the hotspot. Must have the same length as `nplots` (or `nrow(anc)` respectively). See details.
`cf`	Determines the probability of the species to occur on the plots. In other words, it changes the shape of the species accumulation curve. Set to NULL if no natural species accumulation should be applied (may sometimes increase the visibility of the gradient)
`puq`	Percentage of ubiquitous species. Set to NULL if the produced gradients seem to be unclear or if you don't want ubiquitous species to be in the data-set. Only used if a gradient vector is given (which is then not applied to the given percentage of species).

reord	Triggers reordering of the columns in the produced gradient matrix (see details). May considerably change the resulting matrix. Defaults to TRUE.
frac	Numeric between 0 and 1 giving the percentage of species which should occur on the hotspot-gradient only (see details).
n.iter	Number of iterations when ads.fbg is used for finding the species matrix representing best the prescribed gradient (see details).
method	Which method of makead, ads, ads.hot should be used?
...	Further arguments to the function specified in method

Details

There are three different implementations to create an artificial species matrix and a fourth function ads.fbg that allows to use either of the three possibilities to find a "best" gradient.

makead first applies the **n**atural **s**pecies **a**ccumulation curve, the gradient for each species is represented by a vector containing numerics between 0 and 1. Both matrices are added so that values between 0 and 2 result. Through an iteration procedure a breakvalue is defined above which all entries are converted to 1. Values below are converted to 0 resulting in a presence/absence matrix. However the random element seems to be too strong to get evident gradient representations.

Therefore ads is implemented. It works different. First, a gradient is applied. As with makead the gradient is always applied in two directions so that half of the species are more likely to occur on plots on one side of the gradient, whereas the others are more likely to occur on the other side of the gradient. Subsequently, species occurrence for all species will oscillate around nplots/2.

If puq is specified the given percentage of species is divided from the whole matrix before the gradient is applied. With the parameter cf a vector is produced representing quasi-natural occurrence of the species on the plots: Most species are rare and few species are very common. This is described by a power function $y = \frac{1}{x^{cf}}$ with x starting at 2 and gives a vector of length nspec representing the number of times each species is occuring.

These numbers are applied to the gradient matrix and from the species occurrences only as many as specified by the respective number are randomly sampled. In cases were the occurrence number given by the vector exceeds the occurrences resulting from the gradient matrix, the species in the gradient matrix is replaced by a new one for which occurrence is not following the gradient and represents the number of occurrences given by the vector. The idea behind this is, that also in nature a species occuring on more than about half of the plots will likely be independent from a specific gradient.

In both cases (makead and ads) a totally random species matrix (under consideration of natural species occurrence, see cf) is obtained by randomly shuffling these occurrences on the columns (species) of the "natural species occurrence" matrix.

Contrarily to the other two functions, ads.hot allows for the creation of an artficial data-set including a hotspot of species richness and composition. In this case, frac can be used to specify which proportion of the total number of species should only occur on the hotspot gradient. All other species occur randomly on the plots. However, with the hotspot-gradient (grad.v) you can influence the explicitness of the hotspot.

The function ads.fbg allows for finding the best gradient representation with one of the above functions. A gradient is considered to be represented best, when the correlation between the first axis scores of a DCA (which is calculated with decorana of package vegan and the gradient positions as described by the gradient vector grad.v are maximized. ads.fbg just runs the specified makead function n.iter times and gives out the best result matrix and the r2.adjust value that has been obtained.

Value

The three functions for creating an artificial species matrix each return a presence/absence species matrix with rows representing plots/sampling units and columns representing species. `ads.fbg` returns a list with

`mat`	The species matrix as for the three artificial data set functions.
`r2.adj`	The adjusted r2 value for the regression of the first axis DCA scores of the resulting species matrix against the position on the prescribed gradient as described by the gradient vector `grad.v`.

Author(s)

Gerald Jurasinski, Vroni Retzer

Examples

```
## create a random data-set with 200 species on 60 plots
artda <- makead(200, 60, avSR=25)

## create a gradient running from North to South (therefore you
## need a spatially explicit model of your data which is obtained
## with hexgrid())
coor <- hexgrid(0, 4000, 200)
coor <- coor[order(coor$ROW),] #causes coordinates to be in order.
## then the gradient vektor can easily be generated from the ROW names
gradvek <- as.numeric(coor$ROW)
## check how many plots your array has
nrow(coor)
## create a data-set with 200 species
artda <- ads(200, 100, grad.v=gradvek)
## see the species frequency distribution curve
plot(sort(colSums(artda)))
```

`mama` *A (convenience) wrapper function to make matrix from a data.frame*

Description

The function `mama` uses `reshape` to transpose species data given in database list format (where each line represents a species in a plot, so the list has three columns containing information on plot, species and information on occurence) into a plot species matrix (where rows represent plots and columns represent species) for further use with other functions on vegetational data.

Usage

```
mama(dat)
```

Arguments

`dat`	Species data in list format. The columns have to represent plot, species, occurence information (presence/absence or abundances). Column names may differ but they must be in that order!

Details

You could reach the same result with `reshape`. I was just always quite confused with this. That's why i decided to do this little wrapper for convenience. It needs quite a while to run though, but this is due to `reshape`.

Value

Returns a `data.frame` which contains the presence/absence or abundance data of the species list. Rows represent plots, columns represent species. If you want to have it vice versa you have to use the function on a list with columns `species`, `plots`, `occurrence information` (in that order).

Author(s)

Gerald Jurasinski

See Also

`reshape`, `data.frame`

Examples

```
data(abis)
abis.spcls <- liste(abis.spec, splist=TRUE)
## see the list, it like what you get from a database
## and return to matrix-format:
abis.test <- mama(abis.spcls)
```

`occ.time` *Track species occurrence*

Description

Calculate the change in occurence of species on plots in general or on specific plots. Allows for the tracking of temporal changes in species abundance throughout an area or the simple quantification and comparison of species occurences between time steps.

Usage

```
occ.time(x, y, times = NULL, adjust = TRUE, gen.occ = FALSE,
perc = TRUE, nc.acc = FALSE, ...)

occ.tmp(x, y, adjust=TRUE, gen.occ=FALSE, perc=TRUE,
nc.acc=FALSE, ...)
```

Arguments

x	Species data in matrix or database-format representing species occurrence at time step one or throughout a time series. The latter means that you have a table with three columns where the columns represent plots, species and occurence information (in this order!). These are typically exported from a database. When times are given and data represents more than one time steps it has to be in database format. Conversion is handled automatically - if three columns are in the table it is assumed to be in database format. When there are only three species give the data in database-format. If there are many plots and/or species, internal matrix conversion might be quite slow.
y	Species data in matrix or database-format representing species occurrence at time step two. Obsolete when times are given. Otherwise the same as for x applies.
times	When there are more than two timesteps. A vector describing the timesteps which has to be coercible to a factor. If your data comes from a database and contains species records for different time-steps, just export the time information with the species data. If you have single matrices for each time step, you could reshape them to database format via liste and concatenate these or calculate each time step alone.
adjust	Do not change the default behaviour (TRUE) unless you know what you do. Would spare some calculation time if set to FALSE, when your species data do not need adjustment, which means that in both or all time steps, there are exactly the same species and the same plots. However in most cases it will be more convenient to rely on the function (see details).
gen.occ	Triggers if general occurrence is regarded or specific occurrence. The latter is default (gen.occ=FALSE) and it means that it is calculated on which exact plots a species is changing. When set to TRUE only general occurence is regarded and it is calculated on how many plots a species occurs more or less then before. See details.
perc	If output shall be in percentage of species. Defaults to TRUE.
nc.acc	Per default, species which are not changing on a plot are counted as single species (also when they do not change on more than one plot). This can be changed when setting nc.acc = TRUE. Then each occurence of species which has not changed is counted.
...	Further arguments to functions.

Details

If you compare species data among time steps there will be most likely different numbers of species (and often also different numbers of plots for which information is available). The function takes care of this and you can give any species matrices you want. If one plot is the same, it will calculate what changed on this plot. There will be an error message if no plot is shared. The function relies on plot and species names!! As in a database - they must be unique!!

With the resulting named vector or table it is easy to draw a barplot tracking the changing occurrence of plants. Walter et al. (2005) used such plots to illuminate the changing (and increasing) occurrence of plant species on Alpine summits due to climate change.

However they only considered change in general occurrence. We added the possibility to track changes on the specific plots. If a species is occuring on 5 plots at time one and on 4 plots at time two one can't be shure that this species changed occurrence on only one plot. If it occurrs partly

on different plots then before it will contribute to loss and gain respectively in this function when `gen.occ` is set to FALSE.

Value

Returns a named vector or a table (when multiple time steps are evaluated at once) with information on change in species occurence. The names tell on how many plots a certain species has changed. The values tell how many species (or percent of species) exhibit this change. Additionally the number of matching plots and species for each comparison are given back.

Per default nice output is given. However, the table can be accessed for printing whith *\$bac (see Example for Details). A plotting method will be added in the near future.

Author(s)

Gerald Jurasinski <gerald.jurasinski@uni-rostock.de>

References

Walther, G.-R., Beißner, S. & Burga, C. A. 2005. Trends in the upward shift of alpine plants. *Journal of Vegetation Science* **16**: 541-548.

See Also

to calculate similarity based on plant species occurrence between time steps see `sim.tmp`

Examples

```
## load included data
data(bernina)

## how species changed occurrence between first recording
## and last recording?
# construct a species matrix that only contains the species
# that occurred on the summits at the first recording
first <- veg[summits$year=="1907",]
first <- first[,colSums(first)>0]
# make right summit names
row.names(first) <- as.character(summits$summit[summits$year=="1907"])
# construct a species matrix that only contains the species
# that occurred on the summits at the last recording
last <- veg[summits$year=="2003",]
last <- last[,colSums(last)>0]
# make right summit names
row.names(last) <- as.character(summits$summit[summits$year=="2003"])
# see that dimensions of first and last differ
dim(first)
dim(last)
# calculate change in occurrence
changed.occurrence <- occ.tmp(first, last)
barplot(changed.occurrence$bac, main="percentage of species
that changed on ... plots")
# try the same but species are not given in percentages
changed.occurrence <- occ.tmp(first, last, perc=FALSE)
barplot(changed.occurrence$bac, main="number of species that
changed on ... plots")
# there is a lot of info in the output
```

```
changed.occurrence

## how species changed occurrence between recordings?
## data has to come in list format
occ.time(veg.lst, times=years)
```

`pcol` *Permuted Correlation (on strata)*

Description

The function is a wrapper for several functions related to the permuted correlation between matrices or vectors: It calls `permcor2` to calculate permuted correlation between vectors and `permcor` to calculate permuted correlation on strata. This can be useful to obtain data-points for a multivariate Mantel correlogram. Two matrices or vectors and a variable dividing these vectors into strata (levels) are to be specified. If the last is not given only permuted correlation between the two matrices (`dist`-objects) is done. If the second matrix/vector is a subsetting object the correlation can be done on the first object for each of the strata with `mancor`.

Usage

```
pcol(x, y, z = NULL, method="pearson", permutations=1000,
solo=FALSE, ...)

permcor(x, y, subsetter, method="pearson", permutations=1000,
alpha=0.05, trace=FALSE, ...)

permcor2(x, y, method="pearson", permutations=1000, subset=NULL,
complete=TRUE, loop=FALSE, ...)

mancor(dis, classes, width=NULL, method="pearson",
permutations=1000, alpha=0.05, trace=FALSE, ...)

## S3 method for class 'permcor':
plot(x, y, ...)
```

Arguments

`x`	Matrix (`dist`-objekt) or vector of numeric values (atomic) containing distances or similarities returned by `sim`, `vegdist`, `dist`, `dist.binary` or a similar matrix. Conversion is done automatically and triggered by `is.vector`. If it is not a vector it is assumed to be a `dist`-objekt or a similar matrix (with nrow==ncol). For the plotting method a `permcor`-object.
`y`	If `solo`=FALSE an object of the same qualities and dimensions (or length) as `x` to correlate to. See `Details` for the case `solo`=TRUE. For the plotting method an optional object. Not needed here.
`z`	Vector or `dist`-objekt of a variable which defines the strata, or levels for which the permuted correlation shall be calculated. Doesn't have to be a factor but it has to be convertible into factor. see `Details` for more.

subsetter	Vector of a variable which defines the strata, or levels for which the permuted correlation shall be calculated. Usually conversion from dist-object to vector is done in pcol. If permcor is run separately it has to be a vector (same holds for x, y, and z if the background functions are run separately). Doesn't have to be a factor but it has to be convertible into factor. see Details for more.
method	Method of correlation, as it is done by cor.test, see help there for details. Defaults to Pearson correlation coefficients. Other options are Kendall and Spearman rank correlations.
permutations	number of permutations, defaults to 1000 to get a significance level of p = 0.001.
solo	If TRUE y is assumed to be a subsetting matrix, dist-object or vector giving strata to correlate x with and calculate the data-points for a Multivariate Mantel Correlogram for one distance-matrix.
width	If solo=TRUE the subsetting object is assumed to contain classes already. If width is specified (defaults to NULL), the classes are defined inside the function and width gives the class width.
trace	Set to TRUE to follow the runs of the for-loops in functions permcor and mancor. See details for when it is appropriate to set via pcol.
complete	Should only complete cases be considered? Defaults to TRUE.
alpha	The initial alpha-level against which should be tested. Depending on sub-function it is internally changed via Bonferroni-correction if necessary.
subset	If Case is 1 (see details) a subset of cases from x and y can be defined for correlation.
loop	Triggers the method for permutation inside the function. Shall it be looped (for-loop, loop = TRUE) or be done by an apply method (loop = FALSE)? Determines speed. For the most, reasonably huge data sets, the latter will be better.
dis	Same as x for mancor.
classes	If mancor shall be run, second item is a vector or dist-objekt of a variable which defines the strata, or levels for which the permuted correlation shall be calculated. Doesn't have to be a factor but it has to be convertible into factor. see Details for more.
permcor	(In plot.permcor) an object returned by permcor2 is easily plotted with plot.permcor. The actual correlation value is plotted against an histogram of the distribution of the permuted values.
...	Arguments to other functions, for instance to cor.test regarding specifications of the test, however only the correlation value is taken from this function. but here you could change from pearson to kendall for instance.

Details

pcol is a wrapper for the other functions. Depending on the input and the setting of solo the following functions are invoked (They can also be run separately. In this case x, y, and z must be given as vectors).

1. If x and y are dist-objects, or vectors containing distance or similarity values and everything else is set to defaults a simple permuted correlation with permcor2 is run. This corresponds to a Mantel test. The two data-objects are correlated with cor.test, then the y is permuted and with cor the correlation is calculated again and written to a vector. This is

repeated permutation times. Finally, the initial correlation value is compared to the permuted values. The number of times, the permuted values exceed the initial value is divided by the number of permutations to obtain a significance value. Thus, with 1000 permutations a minimum p of 0.001 can be tested.

2. If a subsetter is given in z, the permuted correlation is done for every stratum or level given by the subsetting object - this could e.g. be direction or distance classes flagging which plots share a similar distance and therefore fall into the same class. The resulting data-points can be used to plot a correlogram which allows for the analysis of non-stationarity in the relationships between x and y.
3. If y is itself a subsetting object (distance classes or so) you have to set solo=TRUE. Than the matrix or vector in x is correlated against this classes. This is handled by mancor. The parameter width allows for the calculation of classes inside the function. If for instance a distance-matrix with geographical distances is given, width specifies the width of the distance classes, they are computed and used to correlate the data in x against. This produces the data-points for a multivariate Mantel correlogram in the sense of Oden & Sokal (1986) (see also Legendre & Legendre 1998 for a comprehensive coverage of the subject).

Value

Returns different objects, depending on given arguments and triggers.

case 1 — a permcor-object with the following items is returned:

- call The call to the function.
- method The correlation method as used by cor.test.
- statistic The initial correlation value which is tested against the permuted values.
- signif The significance of the calculation.
- n The number of cases.
- permutations The number of permutations as specified by permutations.
- perms The result of the permuted runs. It is not printed by default but can be accessed via result$perms. The correlation value can be plotted against an histogram of the distribution of the permuted values to visualize the significance with the default plotting method.

case 2 — a pclist-object with the following items (in this case it might be nice to set trace=TRUE to display the progress of the calculation) is returned:

- call The call to the function.
- method The correlation method as used by cor.test.
- gesN The total number of cases.
- strata The number of strata (or levels) for which permutation has been done.
- permutations The number of permutations as specified by permutations.
- out A data.frame with 3 columns containing the result for each stratum in the rows: statistic contains the correlation values for the corresponding stratum, sig the obtained significance, and nop the number of cases found and used for permutation on this very level.

case 3 — a pclist-object with the same items as in Case 2 (in this case it might be as well interesting to set trace=TRUE to display the progress of the calculation) is returned.

Note

Depending on the background-function and the size of the matrices or vectors it may take a while to calculate. The slowest is `mancor` (case 3).

Author(s)

Gerald Jurasinski

References

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

Oden, N. L. & Sokal, R. R. (1986) Directional Autocorrelation: An Extension of Spatial Correlograms to Two Dimensions. *Systematic Zoology* **35**: 608-617.

See Also

`mantel` for a different implementation of Mantel tests, `cor.test`

Examples

```
data(abis)

## calulcate soerensen of species data
abis.soer <- sim(abis.spec)
## calculate distance (Euclidean) regarding some disturbance
## variables (feces counts)
abis.pert <- dist(abis.env[,19:25])
## are compositional similarity and dissimilarity of disturbance related?
pcol(abis.soer, abis.pert)
## the relationship is significant, but not very strong
```

`plot.mrpp` *plot an mrpp-object*

Description

There is no mrpp plotting function in the vegan package. For convenience it is provided here.

Usage

```
## S3 method for class 'mrpp':
plot(x, y, ...)
```

Arguments

`x`	a mrpp-object as computed with `mrpp`
`y`	Optional plotting object, not needed here.
`...`	Arguments to the plotting function

Value

Returns a histogram with the distribution of the permutation values plotted against delta: For explanations see mrpp

Author(s)

Gerald Jurasinski

See Also

mrpp, anosim

---

rin *Calculate multiple plot resemblance measures*

Description

The functions calculate several multiple plot similarity measures. In addition rin provides a wrapper that allows for the easy calculation of multiple plot (site) **r**esemblance measures **i**n **n**eighboorhoods in an automated fashion including testing whether the found resemblance patterns are significantly different from random.

Usage

```
mpd(x, method="simpson", all=FALSE)

mps(x, method="whittaker", all=FALSE)

mps.ave(x, method="soerensen", all=FALSE, foc=NULL,
what="mean", ...)

mps.f(x, foc, d.inc=FALSE, preso=FALSE)

sos(x, method="mean", foc=NULL, normal.sp=TRUE, normal.pl=TRUE)

rin(veg, coord, dn, func, test = TRUE, permutations = 100,
permute = 2, sfno = TRUE, p.level = 0.05, ...)
```

Arguments

x	Species composition data, a matrix-like object.
method	Method for the calculation of multiple plot resemblance. The possible choices depend on the function used and include (among others) Simpson based multiple plot dissimilarity, Sørensen based multiple plot dissimilarity, Nestedness based multiple plot dissimilarity, Whittaker's beta, additive partitioning, Harrison multiple plot similarity, Harrison multiple plot turnover, Williams multiple plot turnover, average pairwise similarity (with a similarity measure of your choice from sim), Diserud & Ødegaard multiple plot similarity. The methods of mps.f (a new group of multiple plot similarity measures) evolve from setting the arguments accordingly. For sos the choice is mean or foc. See details.

`all` Logical. Depending on the function this argument has a different meaning. In `mps` and `mpd` it sets whether the results of all possible methods shall be given in the result, or only the method given in the `method` argument. Because some of the measures are just derived from others all methods are always calculated within the function when it is called and the `method` argument just triggers which to give back. In `mps.ave` it sets whether all statistics calculated (`mean` and `sd`) shall be given back or only the one specified by the `what` argument.

`foc` Character vector with length one or an integer specifying which one is the focal plot. Three of the functions are/can be sensitive to the species composition in the focal plot (`mps.f`, `mps.ave`, `sos`). The automation function `rin` is able to automatically derive the identity of the focal plot. Just set `foc = foc` in the `func` argumemt (see example). When `mps.ave` or `mps.f` are used stand alone either the name of the plot in parenthesis or the index of the plot within the species matrix (`x`) has to be given.

`what` For `mps.ave`, which statistic (`mean` or `sd`) should be given back? See details.

`d.inc` Logical. Shall all species that are within `veg` but not within the plots that make up a neighborhood be regarded when computing `mps.f`. This setting dramatically changes the behaviour of `mps.f` because it then becomes a symmetric similarity coefficient. Defaults to `FALSE` so that an asymmetric multiple plot similarity coefficient is computed. Only makes sense when `mps.f` is applied within `rin` and changes nothing otherwise.

`preso` Logical. Shall a presence only version of `mps.f` be computed? Default is `FALSE`. See details.

`normal.sp` In case of `sos` (**s**um **o**f **s**quares of species matrix, which is a measure of beta-diversity (Legendre et al. 2005)): Shall the result be normalized with respect to the number of species.

`normal.pl` In case of `sos` (**s**um **o**f **s**quares of species matrix, which is a measure of beta-diversity (Legendre et al. 2005)): Shall the result be normalized with respect to the number of plots.

`veg` Species composition data, a matrix-like object that is ought to be recorded in a regular array or a similar structure and that shall be divided into neighborhoods with a moving window so that each plot becomes the focal plot with a certain neighborhood of plots around for which the multiple plot resemblance measures are then calculated.

`coord` Spatial coordinates of the field plots where the data in veg comes from. The function expects a `data.frame` with two columns with the first column giving the x (easting) coordinate and the second giving the y (northing) coordinate in UTM or the like. These coordinates are used to calculate the neighborhoods within a moving window approach.

`dn` Distance to neighbors or neighbor definition. A positive numeric, a two value vector (also positive numeric), or a character string. In the first case it gives the distance from each sampling unit in m until which other sampling units should be seen as neighbours. In the second the two values define a ring around each plot. Plots that fall into the ring are considered as neighbors. In the third case, the character string defines the number of k nearest neighbors that should be regarded as the neighborhood. This being a character just triggers a different way to calculate the neighbors. See details.

`func` A character string that defines the formula which shall be applied to calculate a multiple plot resemblance measure for all possible neighborhoods within an

array. For instance "mpd(x)" to compute the Simpson multiple plot dissimilarity coefficient sensu Baselga (2010). See details.

test — Logical. Shall the significance of the calculated values of multiple plot resemblance be tested regarding its deviation from random expectations. Defaults to TRUE. See details.

permutations — The number of permutations run for testing the significance. Defaults to 100. And it is already slow. So test before you give much higher number of runs here.

permute — When testing with rin, how should the permutation of species to reflect random expectations be done: An integer of either 1, 2, or 3. With 1 the species matrix (veg) is permuted across rows. With 2 the species matrix (veg) is permuted across columns. With 3 the species in the focal plot are permuted (They are randomly drawn from the species pool).

sfno — Species from neigborhood only? Logical, that is only be set in combination with permute = 3. If TRUE, than the species are only drawn at random from the neighboorhod species sub matrix. If set to FALSE, the species are drawn at random from the whole species matrix veg.

p.level — Significance level below which the resemblance patterns shall be considered as significantly different from random expectations. Defaults to 0.05. Enables to give asteriks and stars in the results.

... — Further arguments to the workhorse functions mpd, mps, mps.ave, mps.f can be passed via ....

Details

Several multiple plot similarity indices have been presented that cure some of the problems associated with the approaches for the calculation of compositional similarity for groups of plots by averaging pairwise similarities (Diserud and Ødegaard 2007, Baselga 2010). These indices calculate the similarity between more than two plots whilst considering the species composition on all compared plots. The resulting similarity value is true for the whole group of plots considered (called neighborhood in the following). Further, there are multiple plot similarity coefficients that are determined by the species composition on a reference plot (named focal plot in the following). All of these, can be calculated with the functions described in this help file. See vignette for an overview table. Further, the function rin takes all of them and provides a framework for applying the measures to an array of plots to calculate multiple plot **r**esemblance **i**n **n**eighborhoods (Jurasinski et al. submitted).

mps stands for **m**ultiple **p**lot **s**imilarity, whereas mpd stands for **m**ultiple **p**lot **d**issimilarity; the letters behind the . further specifiy the class of measures that can be calculated with the respective function.

mps.ave calculates **ave**rage multiple plot (dis-)similarities from pairwise (dis-)similarity calculations between the plots in the dataset or in the specified neighborhood. It has several options. With setting the foc argument different from NULL, only the pairwise (dis-)similarities between the specified focal plot and all others in the dataset (neighborhood) are taken to calculate the mean and sd from. When the specified focal plot is not existing, the function will issue a warning and stop. When run with defaults (foc = NULL), all pairwise similarities between the plots in the neighborhood (dataset) are considered. Any resemblance measure available via sim or sim.yo can be taken as base for calculating the average (dis-)similarity and its spread.

mps calculates **m**ultiple **p**lot (dis-)**s**imilarities that are either derived from other approaches to beta-diversity calculation (Whittaker's beta, additive partitioning), or have been around for quite

a while (Harrison multiple plot dissimilarity, Harrison multiple plot turnover, Williams multiple plot turnover). None of these considers the actual species composition on each of the compared plots. The following methods are available (n = number of plots, S = number of species, $\gamma = gammadiversity(S_n)$, $\alpha = alphadiversity(S_i)$):

`whittaker`: Calculates Whittaker's beta (multiplicative partitioning, Whittaker 1960) $\beta = \gamma/mean(\alpha)$.

`inverse.whittaker`: Inverse Whittaker's beta (multiplicative partitioning). Scales between 1/n (when the considered plots do not share any species at all) and 1 (when all plots share the same species)

`additive`: Additive partitioning. Following Lande (1996) and keeping it with α = species number, the additive beta-component of the neighborhood (in the `rin`-case or the complete dataset in the `mps`-case) is calculated.

`harrison`: Harrison (1992) multiple plot dissimilarity. A transformation of Whittaker's beta to be bounded between 0 and 1 ($\frac{\beta_W - 1}{n-1}$.

`diserud`: Diserud & Ødegaard (2007) derived this from the pairwise Sørensen similarity measure. However, as Baselga highlights, this can also be derived from Whittaker's beta $\frac{n-\beta_W}{n-1}$ and is basically the same as Harrisons multiple plot dissimilarity but expressed as a similarity.

`harrison.turnover`: $\frac{\frac{\gamma}{max(\alpha)} - 1}{n-1}$ (Harrison et al. 1992).

`williams`: $1 - \frac{max(\alpha)}{\gamma}$ (Williams 1996).

`mpd` calculates **m**ultiple **p**lot **d**issimilarity indices that have been suggested by Baselga (2010). The following methods are available (The implementation differs slightly from the one offered by Baselga in the electronic appendix of his paper and is computationally more efficient):

`simpson`: mps.Sim in the following. Baselga et al. (2007) derive this multiple plot dissimilarity coefficient directly from the pairwise Simpson dissimilarity index by applying it to a group of plots/sites. The authors emphasize, that this coefficient is independent of patterns of richness and peforms better than the Diserud & Ødegaard cofficient in cases of unequal species numbers between plots, because it discriminates between situations in which shared species are distributed evenly among plots or concentrated in a few pairs of sites.

`sorensen`: mps.Sor in the following. By building multiple site equivalents of the matching components (a, b, c) Baselga (2010) derives a Sørensen based measure of multiple plot dissimilarity.

`nestedness`: mps.nes in the following. Because the Sørensen based multiple plot dissimilarity coefficient accounts for both spatial turnover and nestedness whilst the Simpson based multiple plot dissimilarity coefficient accounts only for spatial turnover, it is possible to calculate the multiple plot similarity that is completely due to nestedness by calculating mps.Sor - mps.Sim.

`mps.f` calculates **f**ocal **m**ultiple **p**lot **s**imilarities. In contrast to the other functions the different outcomes can be triggered by setting the further arguments accordingly.

The indices of `mps.f` are the only ones whose value changes depending on the vegetation composition of the focal plot. It is therefore true and valid only for the comparison of the focal plot with the surrounding plots. Not the similarity in the neighborhood, but the similarity of the focal plot to all others in the neighborhood is calculated. The calculation is based on the occurrences and non-occurrences of species on the compared plots with the species composition on the focal plot determining which of the two is to be used for which species: For all species that occur on the focal plot the proportional frequencies of occurrence in the neighborhood are summed up. For species that do not occur on the focal plot the proportional frequencies of non-occurrence in the neighborhood are summed up.

$$\sum_{i=1}^{s_o} f_{oi} + \sum_{i=1}^{s_n} f_{ni}$$

with f_oi = proportional frequency of occurrences of the ith species on the compared plots, only carried out for species that do occur on the focal plot, f_nj = frequency of non-occurrences of the jth species on the compared plots, only carried out for species that do not occur on the focal plot). The frequencies are calculated against the total numbers of cells in the species matrix and are therefore 'proportional frequencies' (in analogy to 'proportional abundances' as in diversity indices like Shannon or Simpson). Thus, if all compared plots have an identical species composition, the resulting value of the multi-plot similarity coefficient is 1. In this rather hypothetical case the species presence absence matrix would be filled with ones only. This is the null model against which the 'proportional frequencies' are calculated. Therefore, the coefficient can be interpreted as a measure of deviation from complete uniformity. There are three versions.

`preso=TRUE`: In this case a presence only version is calculated (`mps.fpo`). Therefore the second term is skipped and the formula simplifies to $\sum_{i=1}^{s_o} f_{oi}$. This very much glorifies the species composition on the focal plot and evaluates whether the surrounding plots in the neighborhood feature the same species.

`d.inc=FALSE`: When the `d.inc` argument is set to `FALSE`, only the species in the neighborhood build the basis against which the 'proportional frequencies' are calculated. This is the default index `mps.f`.

When run with defaults (`preso = FALSE`) and (`d.inc = TRUE`), a **s**ymmetric **f**ocal **m**ultiple **p**lot **s**imilarity (`mps.fs`) index results. It is definetely meant for use in the context of `rin`. The 'proportional frequencies' are calculated against the whole species matrix. Thus, the index is a symmetric similarity coefficient sensu Legendre & Legendre 1998 that considers species that do not occur on the compared plots but in the whole data set. Therefore, it is more appropriate for biodiversity or conservation studies and not so much for the investigation of ecological relationships. However, it can be interpreted as an 'ordination on the spot': By calculating `mps.fs` for a focal plot against its surrounding plots its position along the main gradient according to its species composition is estimated immediately because the species composition in the rest of the data set is incorporated in the construction of the proportional frequencies of the species. Because of this, `mps.fs` can be interpreted as a measure of deviation from complete unity in species composition. When the neighborhood is increased to the full data set, `mps.f` and `mps.fs` converge.

`sos` calculates the **s**um **o**f **s**quares of a species matrix. Legendre et al. (2005) show, that this is a measure of beta-diversity. However, when you don't normalize against the number of species and/or plots the obtained values can hardly be compared across data sets (or neighborhoods). Therefore, its advisable to run this with defaults (`normal.sp = TRUE` and `normal.pl = TRUE`). For experiments, `method` can be set to `"foc"`. Then, not the deviation from the mean of the species occurence across plots builds the basis, but the deviation from the situation on a focal plot. This makes it somewhat related to the `mps.f`-stuff.

`rin` applies the other functions to an array of plots. For each plot a neighborhood is constructed via the `dn` argument and the specified index is calculated for all plots and neighborhoods. The function to be calculated is specified simply by the `func` argument. For instance, with `func = "mpd(x, method='sorensen')"` the function `rin` calculates the Sørensen multiple plot dissimilarity for each plot and its neighborhood in an array. The functions that need the identity of a focal plot (`mps.ave` and `mps.f`) automatically derive the focal plots. However, to trigger this it has to be specified within the `func` argument: `func = "mps.f(x, foc = foc)"`.

Value

The functions `mpd`, `mps`, `mps.ave`, `mps.f` return a single value with the calculated index (according to the `method` argument, or to the other arguments). When `all` is set to `TRUE`, `mps.ave` returns two values (the average and the standard deviation of the pairwise similarities in the neighborhood), whereas `mpd` and `mps` return a named numerical vector with the values for all indices that can be calculated with the respective function.

`rin` gives back a table (`data.frames`), that reports several values for each plot in the dataset per row. The first three columns are always returned. In case `test = TRUE`, three more columns with information on the significance test are returned.

`n.plots`	Number of plots that make up the neighborhood.
`n.spec`	Number of species that occur in the neighborhood.
`dis`	Value of the calculated (dis)similarity index per plot.
`p.val`	p value of the permutation test. According to the `permute` argument the data set is shuffled. The random data is subjected to the same calculations `permutations` times. The original value of multiple plot similarity is compared to the distribution of random values to obtain this p.
`sig`	Significance flag. Just a translation of the p value into a significance flag. There are only two possibilities: "*" value is significantly different from random, "ns" value is not significantly different from random.
`sig.sign`	The sign of the significance value. The tail which is tested is determined by the relation of the multiple plot similarity value to the average multiple plot similarity value of the random test distribution. Thus, the sign shows whether the multiple plot similarity is significantly higher than can be expected from random expectations (+) of lower (−).

Note

`rin` is not optimized and could perhaps profit from some C code. So when `test = TRUE` it takes a while because of the permutations.

Author(s)

Gerald Jurasinski <gerald.jurasinski@uni-rostock.de>, with contributions by Vroni Retzer <vroni.retzer@gmx.de>

References

Baselga A (2007) A multiple–site similarity measure independent of richness. *Biology Letters* 3: 642–645.

Baselga A (2010) Partitioning the turnover and nestedness components of beta diversity. *Global Ecology and Biogeography* 19: 134–143.

Diserud OH, Ødegaard F (2007) A multiple–site similarity measure. *Biology Letters* 3: 20–22.

Harrison S, Ross SJ, Lawton JH (1992) Beta-diversity on geographic gradients in Britain. *Journal of Animal Ecology* 61: 151–158.

Jurasinski G, Jentsch A, Retzer V, Beierkuhnlein C (submitted) Assessing gradients in species composition with multiple plot similarity coefficients. *Ecography*

Lande R (1996) Statistics and partitioning of species diversity, and similarity among multiple communities. *Oikos* 76: 5–13.

Legendre P, Borcard D, Peres-Neto P (2005) Analyzing beta diversity: partitioning the spatial variation of community composition data. *Ecological Monographs* 75: 435–450.

Williams PH (1996) Mapping variations in the strength and breadth of biogeographic transition zones using species turnover. *Proceedings of the Royal Society of London Series B–Biological Sciences* 263: 579–588.

Whittaker RH (1960) Vegetation of the Siskiyou Mountains, Oregon and California. *Ecological Monographs* 30: 279–338.

See Also

sim, vegdist, dsvdis for pairwise similarity measures.

Examples

```
## Not run:
# load the data that comes with the package
data(abis)

# calculate a multiple plot similarity index
# (Sørensen sensu Baselga) for whole dataset
mpd(abis.spec, method="sorensen")

# calculate a multiple plot similarity index
# (Sørensen sensu Baselga) for each plot and
# its neighborhood
abis.mpd.so <- rin(abis.spec, coord=abis.env[,1:2],
dn=100, func="mpd(x, method='sorensen')")

# plot the grid of plots and show the calculated
# multiple plot dissimilarity value through the
# size of the symbol and the sign of the value
# with a superimposed "+" or "-"
with(abis.mpd.so , {
plot(abis.env[,1:2], cex=symbol.size(dis), pch=c(21,1)[sig],
bg="grey50", xlab="", ylab="")
subs <- sig == "*"
points(abis.env[subs,1:2], pch=c("-", "+")[sig.prefix[subs]])
})

# calculate a multiple plot similarity index
# that takes care of the species composition
# on the focal plot
rin(abis.spec, coord=abis.env[,1:2], test=FALSE,
dn=100, func="mps.f(x, foc=foc)")

## End(Not run)
```

`sim` *Calculate similarities for binary vegetation data*

Description

One of 56 (dis)similarity measures for binary data can be set to calculate (dis)similarities. The vegetational data can be in either database (list) or matrix format. Same holds for the output. Simultaneous calculation of geographical distances between plots and the virtual position of the calculated similarity values between the parental units can be achieved if a `data.frame` with coordinates is given.

Usage

```
sim(x, coord=NULL, method = "soer", dn=NULL, normalize = FALSE,
listin = FALSE, listout = FALSE, ...)
```

Arguments

x	Vegetation data, either as matrix with rows = plots and columns = species (similarities are calculated between rows!), or as data.frame with first three columns representing plots, species and occurence information respectively. All further columns are dumped before calculation. Occurence is only considered as binary. If your list or matrix contains abundances or frequencies they are transformed automatically.
coord	A data.frame with two columns containing the coordinate values of the sampling units. If given, it triggers the simultaneous calculation of the geographical distances between the sampling units, the coordinates of virtual centre-points between all possible pairs of plots, and the geographical distances in either x- or y-direction. If coord is given, output is always in database format (no matrix).
method	Binary Similarity index (see Details for references and formulae), partial match to "soerensen", "jaccard", "ochiai", "mountford", "whittaker", "lande", "wilsonshmida", "cocogaston", "magurran", "harrison", "cody", "williams", "williams2", "harte", "simpson", "lennon", "weiher", "ruggiero", "lennon2", "rout1ledge", "rout2ledge", "rout3ledge", "sokal1", "dice", "kulcz1insky", "kulcz2insky", "mcconnagh", "manhattan", "simplematching", "margaleff", "pearson", "roger", "baroni", "dennis", "fossum", "gower", "legendre", "sokal2", "sokal3", "sokal4", "stiles", "yule", "michael", "hamann", "forbes", "chisquare", "peirce", "eyraud", "simpson2", "legendre2", "fager", "maarel", "lamont", "johnson", "sorgenfrei", "johnson2". See details.
dn	Neighbor definition. A geographic distance represented by a numeric or a two value vector defining a ring around each plot. Only takes effect when coord != NULL. If specified, the output does only contain similarities between neighboring plots. A plot is a neighbour to any given plot if it is within the range of the neighbor definition. See details.
normalize	Logical value indicating whether the values for a, b and c which are calculated in the process should be normalized to 100% (per row, which means per plot comparison). If normalize = TRUE an asymmetric index must be chosen (see details).
listin	if x is given in database (list) format this must be set to TRUE (there is no automatic detection of the format)
listout	If output is wanted in database format rather than as a dist-object set this to TRUE. Output is automatically given in database-format, when coord is specified.
...	Arguments to other functions

Details

All binary similarity indices are based on the variables a, b and c (or can be expressed as such). Some of them also use d. Where a is the number of species shared by two compared plots, b is the number of species found only in one of the compared plots, and c is the number of species only found in the other of the compared plots. d refers to species which are absent from both the compared plots but present in the whole dataset. Indices incorporating d are discussed critically by Legendre & Legendre (1998) and elsewhere. They are called symmetric and expose a "double zero" problem as they take species into account which are absent from both compared units. Absence

of species from a sampling site might be due to various factors, it does not necessarily reflect differences in the environment. Hence, it is preferable to avoid drawing ecological conclusions from the absence of species at two sites (Legendre & Legendre 1998). The indices presented here come from various sources as indicated. Comparative reviews can be found in e.g. Huhta (1979), Wolda (1981), Janson & Vegelius (1981), Shi (1993), Koleff et al. (2003), Albatineh (2006)

The indices considerably differ in their behaviour. For classification purposes and in ecology, Jaccard and Sørensen have been found to give robust and meaningful results (e.g. Janson & Vegelius 1981, Shi 1993). For other purposes other indices might be better suited. However, you are invited to use (at least with the asymmetric indices) ternary plots as suggested by Koleff et al. 2003. The matching components `a, b,` and `c` can be displayed in a ternary.plot to evaluate the position of the plots in similarity space. When output is in database-format, the matching components are always given and `triax.plot` can be used to plot them into a triangle-plot. Koleff et al. (2003) used an artificial set of matching components including all possibilities of values that `a, b,` and `c` can take from 0 to 100 to display the mathematical behavior of indices. An artificial data-set with this properties - together with the values for the asymmetric indices included here - is part of this package (`ads.ternaries`) and can be used to study the behavior of the indices prior to analysis. See details and examples there.

If `coord` is given, the geographic distances between plots/sampling units are calculated automatically, which may be of value when the display or further analyses of distance decay (sensu Tobler 1970, Nekola & White 1999) is in focus. For convenience the `dn`-trigger can be used to tell the function to only return similarities calculated between neighboring plots. Similarities between neighboring plots in an equidistant array are not subjected to the problem of auto-correlation because all plots share the same distance (Jurasinski & Beierkuhnlein 2006). Therefore, any variation occurring in the data are most likely caused by environmental differences alone.

In the following formulae...

`a` = number of shared species

`b` = number of species only found on one of the compared units

`c` = number of species only found on the other of the compared units

`d` = number of species not found on the compared plots but in the dataset

`N` = $a+b+c+d$

with $(n_1 \leq n_2)$...

n_1 = number of species of the plot with fewer species $(a+b)$ or $(a+c)$

n_2 = number of species of the plot with more species $(a+b)$ or $(a+c)$

Computable asymmetric indices:

`soerensen`	$sim = \frac{2a}{2a+b+c}$	Soerensen (1948)
`jaccard`	$sim = \frac{a}{a+b+c}$	Jaccard (1912)
`ochiai`	$sim = \frac{a}{\sqrt{(a+b)(a+c)}}$	Ochiai (1957), Shi (1993)
`mountford`	$sim = \frac{2a}{(a(b+c)+2bc)}$	Mountford (1962), Shi (1993)
`whittaker`	$sim = \frac{a+b+c}{2a+b+c} - 1$	Whittaker (1960), Magurran (1988)
`lande`	$sim = \frac{b+c}{2}$	Lande (1996)
`wilsonshmida`	$sim = \frac{b+c}{2a+b+c}$	Wilson & Shmida (1984)
`cocogaston`	$sim = \frac{b+c}{a+b+c}$	Colwell & Coddington (1948), Gaston et al. (2001)
`magurran`	$sim = (2a+b+c)(1-\frac{a}{a+b+c})$	Magurran (1988)
`harrison`	$sim = \frac{min(b,c)}{max(b,c)+a}$	Harrison et al. (1992), Koleff et al. (2003)
`cody`	$sim = 1 - \frac{a(2a+b+c)}{2(a+b)(a+c)}$	Cody (1993)
`williams`	$sim = \frac{min(b,c)}{a+b+c}$	Williams (1996), Koleff et al. (2003)

`williams2`	$\frac{(bc)+1}{(a+b+c)^2-(a+b+c)}$	Williams (1996), Koleff et al. (2003)
`harte`	$1-\frac{2a}{2a+b+c}$	Harte & Kinzig (1997), Koleff et al. (2003)
`simpson`	$\frac{min(b,c)}{min(b,c)+a}$	Simpson (1949), Koleff et al. (2003)
`lennon`	$\frac{2\lvert b-c\rvert}{2a+b+c}$	Lennon et al. (2001), Koleff et al. (2003)
`weiher`	$sim=b+c$	Weiher & Boylen (1994)
`ruggiero`	$sim=\frac{a}{a+c}$	Ruggiero et al. (1998), Koleff et al. (2003)
`lennon2`	$sim=1-\left[\frac{log\left(\frac{2a+b+c}{a+b+c}\right)}{log2}\right]$	Lennon et al. (2001), Koleff et al. (2003)
`rout1ledge`	$sim=\frac{(a+b+c)^2}{(a+b+c)^2-2bc}-1$	Routledge (1977), Magurran (1988)
`rout2ledge`	$toolong, seebelow$	Routledge (1977), Wilson & Shmida (1984)
`rout3ledge`	$sim=e^{rout2ledge}-1$	Routledge (1977)
`sokal1`	$sim=\frac{a}{a+2(b+c)}$	Sokal & Sneath (1963)
`dice`	$sim=\frac{a}{min((b+a),(c+a))}$	Association index of Dice (1945), Wolda (1981)
`kulcz1insky`	$sim=\frac{a}{b+c}$	Oosting (1956), Southwood (1978)
`kulcz2insky`	$sim=\frac{\frac{a}{2}(2a+b+c)}{(a+b)(a+c)}$	Oosting (1956), Southwood (1978)
`mcconnagh`	$sim=\frac{a^2-bc}{(a+b)(a+c)}$	Hubalek (1982)
`simpson2`	$sim=\frac{a}{a+b}$	Simpson (1960), Shi (1993)
`legendre2`	$sim=\frac{3a}{3a+b+c}$	Legendre & Legendre (1998)
`fager`	$sim=\frac{a}{\sqrt{n_1 n_2}}-\frac{1}{2*\sqrt{n_2}}$	Fager (1957), Shi (1993)
`maarel`	$sim=\frac{2a-(b+c)}{2a+b+c}$	van der Maarel (1969)
`lamont`	$sim=\frac{a}{2a+b+c}$	Lamont and Grant (1979)
`johnson`	$sim=\frac{a}{2b}$	Johnson (1971)
`sorgenfrei`	$sim=\frac{a^2}{(a+b)(a+c)}$	Sorgenfrei (1959)
`johnson2`	$sim=\frac{a}{a+b}+\frac{a}{a+c}$	Johnson (1967)

Computable symmetric indices (including unshared species):

`manhattan`	$sim=\frac{b+c}{a+b+c+d}$	Mean Manhattan, Legendre & Legendre (1998)
`simplematching`	$sim=\frac{a+d}{a+b+c+d}$	Sokal & Michener 1958
`margaleff`	$sim=\frac{a(a+b+c+d)}{(a+b)(a+c)}$	Clifford & Stevenson (1975)
`pearson`	$sim=\frac{ad-bc}{\sqrt{(a+b)(a+c)(d+b)(d+c)}}$	Phi of Pearson, Gower & Legendre (1986), Yule (1912
`roger`	$sim=\frac{a+d}{a+2(b+c)+d}$	Rogers & Tanimoto (1960), Gower & Legendre (1986
`baroni`	$sim=\frac{\sqrt{ad}+a}{\sqrt{ad}+a+b+c}$	Baroni-Urbani & Buser (1976), Wolda (1981)
`dennis`	$sim=\frac{ad-bc}{\sqrt{(a+b+c+d)(a+b)(a+c)}}$	Holliday et al. (2002), Ellis et al. (1993)
`fossum`	$sim=\frac{(a+b+c+d)\left(-\frac{a}{2}\right)^2}{(a+b)(a+c)}$	Holliday et al. (2002), Ellis et al. (1993)
`gower`	$sim=\frac{a-(b+c)+d}{a+b+c+d}$	Gower & Legendre (1986)
`legendre`	$sim=\frac{a}{a+b+c+d}$	Gower & Legendre (1986), Russell/Rao in Ellis et al. (
`sokal2`	$sim=\frac{ad}{\sqrt{(a+b)(a+c)(d+b)(d+c)}}$	Sokal & Sneath (1963)
`sokal3`	$sim=\frac{2a+2d}{(a+d+(a+b+c+d)}$	Sokal & Sneath (1963)
`sokal4`	$sim=\frac{a+d}{b+c}$	Sokal & Sneath (1963)
`stiles`	$sim=log\frac{(a+b+c+d)\left(\lvert ad-bc\rvert-\frac{a+b+c+d}{2}\right)^2}{(a+b)(a+c)(b+d)(c+d)}$	Stiles (1946)
`yule`	$sim=\frac{ad-bc}{ad+bc}$	Yule & Kendall (1973)
`michael`	$sim=\frac{4(ad-bc)}{(a+d)^2+(b+c)^2}$	Michael (1920), Shi (1993)
`hamann`	$sim=\frac{(a+d)-(b+c)}{N}$	Hamann (1961)
`forbes`	$sim=\frac{(aN-2n_2)}{(Nn_1-2n_2)}$	Forbes (1925), Shi (1993)

chisquare	$sim = \frac{(a+b+c+d)(ad-bc)^2}{(a+b)(a+c)(b+d)(c+d)}$	Yule & Kendall (1950)
peirce	$sim = \frac{(ad-bc)}{(a+c)(b+d)}$	Peirce (1884)
eyraud	$sim = \frac{a-(a+b)(a+c)}{(a+b)(a+c)(b+d)(c+d)}$	Eyraud (1936) in Shi (1993)
euclidean	$sim = \frac{\sqrt{b+c}}{a+b+c+d}$	Mean Euclidean in Ellis et al. (1993)
divergence	$sim = \frac{\sqrt{b+c}}{sqrta+b+c+d}$	Ellis et al. (1993)

rout2ledge formula (Routledge, 1977; Koleff et al. 2003):

$$\beta_{R2} = \log(2a+b+c) - \left(\frac{1}{2a+b+c} 2a \log 2\right) - \left(\frac{1}{2a+b+c}((a+b)\log(a+b) + (a+c)\log(a+c))\right)$$

Value

If listout = FALSE a distance matrix of class dist is returned. If listout = TRUE, a data.frame is returned with 7 columns giving the names of the compared plots in the first two and the calculated similarity measure in the third column. The rest of the columns give the values for a, b, c, and d (in this order). Naming of the first three columns can be changed but defaults to NBX (one of the compared plots), NBY (the other one), used index (the values of the calculated index). If coord != NULL, the following columns are given in addition and the columns a:d shift to the end of the data.frame.

distance	Geographical distance between compared plots
X	For plotting purposes, the x-coordinate of the virtual position of the calculated similarity value in the center between the two compared plots
Y	For plotting purposes, the y-coordinate of the virtual position of the calculated similarity value in the center between the two compared plots
xdist	Geographical distance between compared plots, on the x-axis only
ydist	Geographical distance between compared plots, on the y-axis only

Note

In general, concepts of data-handling are taken from vegdist and the calculation of a, b, c and d is taken from dist.binary. Thanks to Jari Oksanen for his vegan package. The indices were collected from the literature and are applicable in different fields of research.

Author(s)

Gerald Jurasinski <gerald.jurasinski@uni-rostock.de>

References

Albatineh, A. N., Niewiadomska-Bugaj, M. & Mihalko, D. (2006) On Similarity Indices and Correction for Chance Agreement. *Journal of Classification* **V23**: 301-313.

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

Clifford, H. T. & Stephenson, W. (1975) *An introduction to numerical classification.* Academic Press, New York, San Francisco, London.

Cody, M. L. (1993) Bird diversity components within and between habitats in Australia. - In: Ricklefs, R. E. & Schluter, D. (eds.), *Species Diversity in Ecological Communities: historical and geographical perspectives*, pp. 147-158, University of Chicago Press, Chicago

Colwell, R. K. & Coddington, J. A. (1994) Estimating terrestrial biodiversity through extrapolation. *Philosophical Transactions of the Royal Society of London Series B-Biological Sciences* **345**: 101-118.

Dice, L. R. (1945) Measures of the amount of ecological association between species. *Ecology* **26**: 297-302.

Ellis, D., Furner-Hines, J., Willett, P. (1993) Measuring the degree of similarity between objects in text retrieval systems. *Perspectives in Information Management* **3**(2): 128-149

Fager, E. W. (1957) Determination and analysis of recurrent groups. *Ecology* **38**: 586-595.

Faith, D. P., Minchin, P. R. & Belbin, L. (1987) Compositional dissimilarity as a robust measure of ecological distance. *Plant Ecology* **69**: 57-68.

Gaston, K. J., Rodrigues, A. S. L., van Rensburg, B. J., Koleff, P. & Chown, S. L. (2001) Complementary representation and zones of ecological transition. *Ecology Letters* **4**: 4-9.

Gower, J. C. & Legendre, P. (1986) Metric and Euclidean properties of dissimilarity coefficients. *Journal of Classification* **3**: 5-48.

Hajdu, L. J. (1981) Graphical comparison of resemblance measures in phytosociology. *Plant Ecology* **V48**: 47-59.

Harrison, S., Ross, S. J. & Lawton, J. H. (1992) Beta diversity on geographic gradients in Britain. *Journal of Animal Ecology* **61**: 151-158.

Harte, J. & Kinzig, A. (1997) On the implications of species-area relationships for endemism, spatial turnover and food web patterns. *Oikos* **80**.

Holliday, J. D., Hu, C.-Y. & Willett, P. (2002) Grouping of Coefficients for the Calculation of Inter-Molecular Similarity and Dissimilarity using 2D Fragment Bit-Strings. *Combinatorial Chemistry & High Throughput Screening* **5**: 155-166.

Hubalek, Z. (1982) Coefficients of association and similarity, based on binary (presence-absence) data: An evaluation. *Biological Reviews of the Cambridge Philosophical Society* **57**: 669-689.

Huhta, V. (1979) Evaluation of different similarity indices as measures of succession in arthropod communities of the forest floor after clear-cutting. *Oecologia* **V41**: 11-23.

Jaccard, P. (1901) Etude comparative de la distribution florale d'une portion des Alpes et du Jura. *Bulletin de la Societé Vaudoise des Sciences Naturelles* **37**: 547-579.

Jaccard, P. (1912) The distribution of the flora of the alpine zone. *New Phytologist* **11**: 37-50.

Johnson, J. G. (1971) A quantitative approach to faunal province analysis. *American Journal of Science* **270**: 257-280.

Johnson, S. C. (1967) Hierarchical clustering schemes. *Psychometrika* **32**: 241-254.

Jurasinski, G. & Beierkuhnlein, C. (2006) Spatial patterns of biodiversity - assessing vegetation using hexagonal grids. *Proceedings of the Royal Irish Academy - Biology and Environment* **106B**: 401-411.

Jurasinski, G. & Beierkuhnlein, C. (submitted) Distance decay and non-stationarity in a semi-arid Mediterranean ecosystem. *Journal of Vegetation Science*.

Koleff, P., Gaston, K. J. & Lennon, J. J. (2003) Measuring beta diversity for presence-absence data. *Journal of Animal Ecology* **72**: 367-382.

Lamont, B. B. & Grant, K. J. (1979) A comparison of twenty-one measures of site dissimilarity. - In: Orlóci, L., Rao, C. R. & Stiteler, W. M. (eds.), *Multivariate Methods in Ecological Work*, pp. 101-126, Int. Coop. Publ. House, Fairland, MD

Lande, R. (1996) Statistics and partitioning of species diversity and similarity along multiple communities. *Oikos* **76**: 25-39.

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

Lennon, J. J., Koleff, P., Greenwood, J. J. D. & Gaston, K. J. (2001) The geographical structure of British bird distributions: diversity, spatial turnover and scale. *J Anim Ecology* **70**: 966-979.

Magurran, A. E. (1988) *Ecological Diversity and its Measurement*. Chapman & Hall, London.

Mountford, M. D. (1962) An index of similarity and its application to classification problems. - In: Murphy, P. W. (ed.) *Progress in Soil Zoology*, pp. 43-50, Butterworths

Ochiai, A. (1957) Zoogeographical studies on the soleoid fishes found in Japan and its neighbouring regions. *Bulletin of the Japanese Society of Fisheries Science* **22**(9): pp. 526-530

Oosting, H. J. (1956) *The study of plant communities: an introduction to plant ecolog*. W. H. Freeman, San Francisco.

Rogers, D. J. & Tanimoto, T. T. (1960) A computer program for classifying plants. *Science* **132**: 1115-1118.

Routledge, R. D. (1977) On Whittaker's components of diversity. *Ecology* **58**: 1120-1127.

Ruggiero, A., Lawton, J. H. & Blackburn, T. M. (1998) The geographic ranges of mammalian species in South America: spatial patterns in environmental resistance and anisotropy. *Journal of Biogeography* **25**: 1093-1103.

Shi, G. R. (1993) Multivariate data analysis in palaeoecology and palaeobiogeography–a review. *Palaeogeography, Palaeoclimatology, Palaeoecology* **105**: 199-234.

Simpson, E. H. (1949) The measurement of diversity. *Nature* **163**: 688.

Simpson, G. G. (1960) Notes on the measurement of faunal resemblance. *American Journal of Science* **258-A**: 300-311.

Sokal, R. R. & Michener, C. D. (1958) A statistical method for evaluating systematic relationships. *University of Kansas Science Bulletin* **38**: 1409-1438.

Sokal, R. R. & Sneath, P. H. A. (1963) *Principles of numerical taxonomy*. W. H. Freeman, San Francisco.

Sørensen, T. (1948) A method of establishing groups of equal amplitude in plant sociology based on similarity of species content. *Biologiske Skrifter* **5**: 1-34.

Sorgenfrei, T. (1959) Molluscan assemblages from the marine middle Miocene of South Jutland and their environments. *Danmark Geologiske Undersøgelse. Serie 2* **79**: 403-408.

Southwood, T. S. (1978) *Ecological Methods*. Chapman and Hall, London.

Stiles (1961) The association factor in information retrieval. *Journal of the Association for Computing Machinery.* **8** 271-279

Weiher, E. & Boylen, C. W. (1994) Patterns and prediction of alpha and beta diversity of aquatic plants in Adirondack (New York) lakes. *Canadian Journal of Botany-Revue Canadienne De Botanique* **72**: 1797-1804.

Whittaker, R. H. (1960) Vegetation of the Siskiyou Mountains, Orgeon and California. *Ecological Monographs* **30**: 279-338.

Williams, P. H. (1996) Mapping variations in the strength and breadth of biogeographic transition zones using species turnover. *Proceedings of the Royal Society of London Series B-Biological Sciences* **263**: 579-588.

Williams, P. H., Klerk, H. M. & Crowe, T. M. (1999) Interpreting biogeographical boundaries among Afrotropical birds: spatial patterns in richness gradients and species replacement. *J Biogeography* **26**: 459-474.

Wilson, M. V. & Shmida, A. (1984) Measuring beta-diversity with presence-absence data. *Journal of Ecology* **72**: 1055-1064.

Wolda, H. (1981) Similarity indices, sample size and diversity. *Oecologia* **50**: 296-302.

Yule, G. U. & Kendall, M. G. (1973) *An introduction to the theory of statistics*. Griffin, London.

Yule, G. U. (1912) *On the methods of measuring association between two attributes*. Journal of the Royal Statistical Society **75**(6): 579-642

See Also

vegdist, dist.binary, dsvdis, dist

Examples

```
data(abis)
##calculate jaccard similarity and output as dist-object
jacc.dist <- sim(abis.spec, method="jaccard")

##calculate Whittaker similarity (with prior normalisation) and
##output as data.frame
whitt.list <- sim(abis.spec, method="whittaker", normalize=TRUE,
listout=TRUE)

##calculate similarity from a database list after Harte & Kinzig (1997)
##and output as dist-object
abis.spec.ls <- liste(abis.spec, splist=TRUE)
hart.dist <- sim(abis.spec.ls, method="harte", listin=TRUE)

## calculate the geographic distances between sites simultaneously
## and return only similarities calculated between neighboring plots
abis.soer <- sim(abis.spec, coord=abis.env[,1:2], dn=100)

## in an equidistant array
## you can plot this nice between the original positions of the
## sites (with the size of the dots expressing number of species
## for the sites, and value of the Sørensen coefficient in between)
require(geoR)
points.geodata(coord=abis.env[,1:2], data=abis.env$n.spec,
cex.min=1, cex.max=5)
points.geodata(coord=abis.soer[,5:6], data=abis.soer$soerensen,
cex.min=1, cex.max=5, col="grey50", add=TRUE)
```

---

sim.tmp *Calculate binary similarity in time*

Description

The function applies one of 56 similarity measures for binary data to calculate compositional similarity of plots between time steps.

Usage

```
sim.tmp(x, y, method = "soer", normalize = FALSE, adjust = TRUE, ...)
```

Arguments

x	Vegetation data, either as matrix with rows = plots and columns = species, or as data.frame with first three columns representing plots, species and occurence information respectively. All further columns are dumped before calculation. Occurence is only considered as binary. If your list or matrix contains abundances or frequencies they are transformed automatically.
y	Same as x for time-step two.
method	One of 42 similarity measures for binary data. The function uses the same indices as sim. See details there. Per default soerensen similarity is calculated.
normalize	Logical value indicating whether the values for a, b and c which are calculated in the process should be normalized to 100% (per row, which means per plot comparison). If normalize = TRUE an asymmetric index must be chosen (for details see sim).
adjust	Do not change the default behaviour (TRUE) unless you know what you do. Would spare some calculation time if set to FALSE, when your species data do not need adjustment, which means that in both or all time steps, there are exactly the same species and the same plots. However in most cases it will be more convenient to rely on the function (see details).
...	Other arguments to sim

Details

If you compare species data among time steps there will be most likely different numbers of species (and often also different numbers of plots for which information is available). The function takes care of this and you can give any species matrices you want. If one plot is the same, it will calculate what changed on this plot. There will be an error message if no plot is shared. The function relies on plot and species names!! As in a database - they must be unique!!

Value

Returns a named vector with the similarities for each site between time steps for each plot.

Author(s)

Gerald Jurasinski <gerald.jurasinski@uni-rostock.de>

References

See references in sim

See Also

See Also as sim (where you can find a much more elaborate see also section as well).

Examples

```
data(bernina)
## load included data
data(bernina)

## how species changed occurrence between first recording
```

```
## and last recording?
# construct a species matrix that only contains the species
# that occurred on the summits at the first recording
first <- veg[summits$year=="1907",]
first <- first[,colSums(first)>0]
# make right summit names
row.names(first) <- as.character(summits$summit[summits$year=="1907"])
# construct a species matrix that only contains the species
# that occurred on the summits at the last recording
last <- veg[summits$year=="2003",]
last <- last[,colSums(last)>0]
# make right summit names
row.names(last) <- as.character(summits$summit[summits$year=="2003"])
# calculate similarity between time steps
sim.tmp(first, last)
```

`sim.yo` *Calculate a binary similarity index you define.*

Description

Enables the calculation of any binary similarity index via the provision of a formula.

Usage

```
sim.yo(x, coord = NULL, method = "(2*a)/((2*a) + b + c)", dn = NULL,
normalize = FALSE, listin = FALSE, listout = FALSE, ...)
```

Arguments

`x`	Vegetation data, either as matrix with rows = plots and columns = species (similarities are calculated between rows!), or as `data.frame` with first three columns representing plots, species and occurence information respectively. All further columns are dumped before calculation. Occurence is only considered as binary. If your list or matrix contains abundances or frequencies they are transformed automatically.
`coord`	A `data.frame` with two columns containing the coordinate values of the sampling units. If given, it triggers the simultaneous calculation of the geographical distances between the sampling units, the coordinates of virtual centre-points between all possible pairs of plots, and the geographical distances in either x- or y-direction. If `coord` is given, output is always in `database` format (no matrix).
`method`	Give the formula for a binary similarity index. Defaults to the formula of Sørensen index. See `sim` for more examples and general explanations.
`dn`	Neighbor definition. A geographic distance represented by a numeric or a two value vector defining a ring around each plot. Only takes effect when `coord` != NULL. If specified, the output does only contain similarities between neighboring plots. A plot is a neighbour to any given plot if it is within the range of the neighbor definition. See details.

normalize	Logical value indicating whether the values for a, b and c which are calculated in the process should be normalized to 100% (per row, which means per plot comparison). If normalize = TRUE an asymmetric index must be chosen (see details in sim).
listin	if x is given in database (list) format this must be set to TRUE (there is no automatic detection of the format)
listout	If output is wanted in database format rather than as a dist-object set this to TRUE. Output is automatically given in database-format, when coord is specified.
...	Arguments to other functions

Details

Presumably this function will rarely be used because sim already allows for the calculation of a large variety of binary similarity coefficients. But just in case you found or thought of an alternative this function is provided. For details regarding similarity indices see sim. You have to give your formula in quotation marks like this: "(2*a)/((2*a) + b + c)".

Value

If listout = FALSE a distance matrix of class dist is returned. If listout = TRUE, a data.frame is returned with 7 columns giving the names of the compared plots in the first two and the calculated similarity measure in the third column. The rest of the columns give the values for a, b, c, and d (in this order). Naming of the first three columns can be changed but defaults to NBX (one of the compared plots), NBY (the other one), used index (the values of the calculated index). If coord != NULL, the following columns are given in addition and the columns a:d shift to the end of the data.frame.

distance	Geographical distance between compared plots
X	For plotting purposes, the x-coordinate of the virtual position of the calculated similarity value in the center between the two compared plots
Y	For plotting purposes, the y-coordinate of the virtual position of the calculated similarity value in the center between the two compared plots
xdist	Geographical distance between compared plots, on the x-axis only
ydist	Geographical distance between compared plots, on the y-axis only

Note

In general, concepts of data-handling are taken from vegdist and the calculation of a, b, c and d is taken from dist.binary. Thanks to Jari Oksanen for his vegan package and the idea to provide a custom build distance formula.

Author(s)

Gerald Jurasinski <gerald.jurasinski@uni-rostock.de>

References

Sørensen, T. (1948) A method of establishing groups of equal amplitude in plant sociology based on similarity of species content. *Biologiske Skrifter* **5**: 1-34.

See Also

vegdist, designdist, dist.binary, dsvdis, dist for other dissimilarity coefficients, and sim for a variety of formulae for similarity coefficients.

Examples

```
data(abis)

##calculate the default Sørensen index
abis.soer <- sim.yo(abis.spec)

##calculate a custom similarity index
abis.sim <- sim.yo(abis.spec, method="(2*a)/((a) + b + c)")
```

---

simbadocs *Display Package Documentation*

Description

Display package documentation using the pager or pdfviewer defined in options.

Usage

```
simbadocs(doc = c("NEWS", "ChangeLog", "mps-coefficients.pdf"))
```

Arguments

doc	The name of the document to view (partial match, case sensitive).

Note

Since R does not have this facility Jari Oksanen provided a workaround for his vegan package (vegandocs). This here is just adapted to simba.

Author(s)

Jari Oksanen, adapted by Gerald Jurasinski

See Also

vignette.

Examples

```
## Not run:
simbadocs("Change")

## End(Not run)
```

`symbol.size` *Little helper function to obtain bubble plots with simple plot and points methods*

Description

Can be used within the cex argument of a plotting command (`plot` or `points`) to scale the symbols with regard to the values of a numeric variable. Thus, a bubble plot is computed.

Usage

```
symbol.size(x, cex.min = 0.2, cex.max = 5)
```

Arguments

`x`	A numeric vector giving the data for scaling the symbols in a scatter (however, you can scale everything that's got a `cex` argument). The vector must have the same length as `x` (and `y`) that give the plotting positions.
`cex.min`	The minimum size (scaling) of the plotted symbols.
`cex.max`	The maximum size (scaling) of the plotted symbols.

Details

The same, but in a mapping context can be achieved by `bubble`. Also `symbols` provides a comparable result. However, with `symbol.size` a very simple interface to bubble plots that can be used within regular scatter plot methods is given.

Value

A vector is returned with values between `cex.min` and `cex.max` that represent the values in the input vector x. After all, there is just a scaling applied. The vector is meant to be used to specifiy `cex` whereever appropriate.

Author(s)

Gerald Jurasinski

References

Tufte ER (1998) The visual display of quantitative information. Graphics Press. Cheshire, Connecticut.

See Also

`bubble`, `symbols`

Examples

```
# load abisko data that comes with simba
data(abis)

# take the environmental data end plot species richness at the field plot positions
with(abis.env, {plot(X, Y, cex=symbol.size(n.spec))})

# make kind of a multivariate plot in 3d:
# the relation between shannon, evenness, and simpson index (bubble size)
with(abis.env, {plot(shannon, even, cex=symbol.size(simps, cex.max=8))})
```

Index

∗Topic **aplot**
symbol.size, 55
∗Topic **array**
rin, 38
∗Topic **datagen**
hexgrid, 25
makead, 27
∗Topic **datasets**
abis, 2
ads.ternaries, 4
bernina, 10
∗Topic **documentation**
simbadocs, 55
∗Topic **dplot**
symbol.size, 55
∗Topic **hplot**
boxes, 11
∗Topic **htest**
pcol, 33
rin, 38
∗Topic **iplot**
symbol.size, 55
∗Topic **manip**
bb2num, 8
liste, 26
mama, 30
∗Topic **math**
auc, 6
∗Topic **methods**
aslopect, 5
bcoov, 9
com.sim, 12
dfcor, 14
diffmean, 16
diffmich, 18
diffslope, 19
direct, 22
dist.tmp, 23
hexgrid, 25
occ.time, 31
pcol, 33
plot.mrpp, 37
sim, 44
sim.tmp, 51
sim.yo, 53
∗Topic **models**
auc, 6
∗Topic **multivariate**
com.sim, 12
dfcor, 14
dist.tmp, 23
occ.time, 31
rin, 38
sim, 44
sim.tmp, 51
sim.yo, 53
∗Topic **package**
simba-package, 1
∗Topic **ts**
auc, 6
∗Topic **univar**
aslopect, 5
bcoov, 9
diffmean, 16
diffmich, 18
diffslope, 19
∗Topic **utilities**
bb2num, 8
simbadocs, 55

abis, 2
ads, *27*
ads *(makead)*, 27
ads.ternaries, 4, *46*
anosim, *37*
apply, *35*
aslopect, 5
auc, 6

bb2num, 8
bcoov, 9
bernina, 10
boxes, 11
boxplot, *11, 12*
boxplot.n, *11, 12*
bubble, *56*

com.sim, 12
cor, *14, 15*, *35*
cor.test, *2*, *23*, *34–36*

data.frame, *19, 20*, *22, 23*, *27*, *30*, *44*, *48*, *51*, *53, 54*
decorana, *29*
designdist, *54*
dfcor, 14
diffic *(diffslope)*, 19
diffmean, *13*, 16
diffmich, *17*, 18
diffslope, *17*, 19
diffslope2 *(diffslope)*, 19
direct, 22
direct2 *(direct)*, 22
dist, *2*, *5*, *23, 24*, *26*, *33–35*, *45*, *48*, *50*, *53, 54*
dist.binary, *2*, *26*, *34*, *48*, *50*, *54*
dist.quant, *9*
dist.tmp, 23
dsvdis, *2*, *9*, *43*, *50*, *54*

fitmich, *18, 19*
fitmich *(diffmich)*, 18

hexgrid, 25

is.vector, *34*

liste, 26, *31*
lm, *18*, *20, 21*

makead, 27
mama, 30
mancor, *15*, *33*
mancor *(pcol)*, 33
mantel, *15*, *23*, *36*
mean, *38*, *40*
mpd *(rin)*, 38
mps *(rin)*, 38
mrpp, *12*, *14*, *37*

nls, *19*

occ.time, 31
occ.tmp *(occ.time)*, 31
options, *55*

pcol, *15*, 33
permcor, *15*, *33*
permcor *(pcol)*, 33
permcor2, *14, 15*, *33*
permcor2 *(pcol)*, 33
plot, *12*
plot.diffmich *(diffmich)*, 18
plot.dmn *(diffmean)*, 16
plot.dsl *(diffslope)*, 19
plot.mrpp, 37
plot.permcor *(pcol)*, 33
predict, *7*

reshape, *2*, *27*, *30*
rin, 38

sample, *19*, *21*
sd, *38*, *40*
sim, *4*, *13*, *24*, *26*, *34*, *38*, *40*, *43*, 44, *51–54*
sim.tmp, *32*, 51
sim.yo, *40*, 53
simba *(simba-package)*, 1
simba-package, 1
simbadocs, 55
sos *(rin)*, 38
spsample, *25*
summits *(bernina)*, 10
symbol.size, 55
symbols, *56*

triax.plot, *45*

veg *(bernina)*, 10
vegandocs, *55*
vegdist, *2*, *9*, *13*, *24*, *26*, *34*, *43*, *48*, *50*, *54*
vignette, *55*

years *(bernina)*, 10