Package ‘ecodist’

July 13, 2017

Version 2.0.1

Date 2017-07-12

Title Dissimilarity-Based Functions for Ecological Analysis

Author Sarah Goslee [aut, cre],
Dean Urban [aut]

Maintainer Sarah Goslee <Sarah.Goslee@ars.usda.gov>

Depends R (>= 3.0.0)

LazyData true

Imports stats, graphics

Suggests knitr, testthat

VignetteBuilder knitr

Description Dissimilarity-based analysis functions including ordination and Mantel test functions, intended for use with spatial and community data.

License GPL (>= 2)

NeedsCompilation yes

Repository CRAN

Date/Publication 2017-07-13 05:25:44 UTC

R topics documented:

ecodist-package 2
addord 3
bcdist 6
bump 7
bump.pmgram 8
cor2m 9
corgen 10
crosstab 11
distance 12
fixdmat 14
full 15

graze 16
iris.fit 18
iris.nmds 19
iris.vf 20
iris.vfrot 21
lower 23
mantel 24
mgram 26
mgroup 28
min.nmds 30
MRM 31
nmds 33
pco 35
plot.mgram 36
plot.nmds 38
plot.vf 39
pmgram 40
residuals.mgram 43
vf 45
xdistance 46
xmantel 48
xmgram 50
z.no 52
z.z1 54

Index **56**

`ecodist-package` *Dissimilarity-Based Functions for Ecological Analysis*

Description

Dissimilarity-based analysis functions including ordination and Mantel test functions, intended for use with spatial and community data.

Details

This package contains well-established dissimilarity-based ecological analyses, such as `nmds` and `mantel`, and experimental/research analyses such as `xmantel`. Helper functions such as `crosstab` and `cor2m` facilitate analysis of community data.

Because many of the analyses are time-consuming, this package includes worked examples that can be loaded using `data()`.

Index of help topics:

```
MRM                     Multiple Regression on distance Matrices
addord                  Fit new points to an existing NMDS
                        configuration.
```

bcdist	Bray-Curtis distance
bump	Nine-bump spatial pattern
bump.pmgram	Nine-bump spatial pattern
cor2m	Two-matrix correlation table
corgen	Generate correlated data
crosstab	Data formatting
distance	Calculate dissimilarity/distance metrics
ecodist-package	Dissimilarity-Based Functions for Ecological Analysis
fixdmat	Distance matrix conversion
full	Full symmetric matrix
graze	Site information and grazed vegetation data.
iris.fit	Example of adding to an ordination
iris.nmds	Example for nmds
iris.vf	Example for vector fitting on ordination
iris.vfrot	Example for vector fitting on rotated ordination
lower	Lower-triangular matrix
mantel	Mantel test
mgram	Mantel correlogram
mgroup	Mantel test for groups
min.nmds	Find minimum stress configuration
nmds	Non-metric multidimensional scaling
pco	Principal coordinates analysis
plot.mgram	Plot a Mantel correlogram
plot.nmds	Plot information about NMDS ordination
plot.vf	Plots fitted vectors onto an ordination diagram
pmgram	Partial Mantel correlogram
residuals.mgram	Residuals of a Mantel correlogram
vf	Vector fitting
xdistance	Cross-distance between two datasets.
xmantel	Cross-Mantel test
xmgram	Cross-Mantel correlogram
z.no	Example for pmgram
z.z1	Example for pmgram

Author(s)

Sarah Goslee and Dean Urban

Maintainer: Sarah Goslee <Sarah.Goslee@ars.usda.gov>

`addord` *Fit new points to an existing NMDS configuration.*

Description

Uses a brute force algorithm to find the location for each new point that minimizes overall stress.

Usage

```
addord(origconf, fulldat, fulldist, isTrain, bfstep = 10, maxit = 50, epsilon = 1e-12)
```

Arguments

origconf	The original ordination configuration.
fulldat	The dataset containing original and new points.
fulldist	A dissimilarity matrix calculated on `fulldat`.
isTrain	A boolean vector of length `nrow(fulldat)` indicating which rows were training data used in determining `origconf` (TRUE), or are new points (FALSE).
bfstep	A tuning parameter for the brute force algorithm describing the size of grid to use.
maxit	The maximum number of iterations to use.
epsilon	Tolerance value for convergence.

Details

A region comprising the original ordination configuration plus one standard deviation is divided into a grid of `bfstep` rows and columns. For a new point, the grid cell with the lowest stress is identified. That cell is divided into a finer grid, and the lowest-stress cell identified. This process is repeated up to `maxit` times, or until stress changes less than `epsilon`.

Value

fullfitconf	The new ordination configuration containing training and new points.
stress	The stress value for each point.
isTrain	The boolean vector indicating training set membership, for reference.

Author(s)

Sarah Goslee

Examples

```
data(iris)
iris.d <- dist(iris[,1:4])

### nmds() is timeconsuming, so this was generated
### in advance and saved.
### set.seed(1234)
### iris.nmds <- nmds(iris.d, nits=20, mindim=1, maxdim=4)
### save(iris.nmds, file="ecodist/data/iris.nmds.rda")
data(iris.nmds)

# examine fit by number of dimensions
plot(iris.nmds)

# choose the best two-dimensional solution to work with
```

```
iris.nmin <- min(iris.nmds, dims=2)

# rotate the configuration to maximize variance
iris.rot <- princomp(iris.nmin)$scores

# rotation preserves distance apart in ordination space
cor(dist(iris.nmin), dist(iris.rot))

# fit the data to the ordination as vectors
### vf() is timeconsuming, so this was generated
### in advance and saved.
### set.seed(1234)
### iris.vf <- vf(iris.nmin, iris[,1:4], nperm=1000)
### save(iris.vf, file="ecodist/data/iris.vf.rda")
data(iris.vf)

# repeat for the rotated ordination
### vf() is timeconsuming, so this was generated
### in advance and saved.
### set.seed(1234)
### iris.vfrot <- vf(iris.rot, iris[,1:4], nperm=1000)
### save(iris.vfrot, file="ecodist/data/iris.vfrot.rda")
data(iris.vfrot)

par(mfrow=c(1,2))
plot(iris.nmin, col=as.numeric(iris$Species), pch=as.numeric(iris$Species), main="NMDS")
plot(iris.vf)
plot(iris.rot, col=as.numeric(iris$Species), pch=as.numeric(iris$Species), main="Rotated NMDS")
plot(iris.vfrot)

####### addord example

# generate new data points to add to the ordination
# this might be new samples, or a second dataset

iris.new <- structure(list(Sepal.Length = c(4.6, 4.9, 5.4, 5.2, 6, 6.5, 6,
6.8, 7.3), Sepal.Width = c(3.2, 3.5, 3.6, 2.3, 2.8, 3, 2.7, 3.1,
3.2), Petal.Length = c(1.2, 1.5, 1.5, 3.5, 4.1, 4.2, 4.8, 5,
5.7), Petal.Width = c(0.26, 0.26, 0.26, 1.2, 1.3, 1.4, 1.8, 2,
2), Species = structure(c(1L, 1L, 1L, 2L, 2L, 2L, 3L, 3L, 3L), .Label = c("setosa",
"versicolor", "virginica"), class = "factor")), .Names = c("Sepal.Length",
"Sepal.Width", "Petal.Length", "Petal.Width", "Species"), class = "data.frame", row.names = c(NA,
-9L))

# provide a dist object containing original and new data
# provide a logical vector indicating which samples were used to
# construct the original configuration

iris.full <- rbind(iris, iris.new)
all.d <- dist(iris.full[,1:4])
is.orig <- c(rep(TRUE, nrow(iris)), rep(FALSE, nrow(iris.new)))

### addord() is timeconsuming, so this was generated
```

```
### in advance and saved.
### set.seed(1234)
### iris.fit <- addord(iris.nmin, iris.full[,1:4], all.d, is.orig, maxit=100)
### save(iris.fit, file="ecodist/data/iris.fit.rda")
data(iris.fit)

plot(iris.fit$conf, col=iris.full$Species, pch=c(18, 4)[is.orig + 1], xlab="NMDS 1", ylab="NMDS 2")
title("Demo: adding points to an ordination")
legend("bottomleft", c("Training set", "Added point"), pch=c(4, 18))
legend("topright", levels(iris$Species), fill=1:3)
```

---

bcdist *Bray-Curtis distance*

Description

Returns the Bray-Curtis (also known as Sorenson, 1 - percent similarity) pairwise distances for the objects in the data. It is duplicated by functionality within `distance` but remains for backward compatibility and because it is substantially faster.

Usage

```
bcdist(x, rmzero = FALSE)
```

Arguments

`x`	matrix or data frame with rows as samples and columns as variables (such as species). Distances will be calculated for each pair of rows.
`rmzero`	If rmzero=TRUE, empty rows will be removed from the data before distances are calculated. Otherwise, the distance between two empty rows is assumed to be 0 (the default).

Value

This function returns a column-order lower-triangular distance matrix. The returned object has an attribute, Size, giving the number of objects, that is, nrow(x). The length of the vector that is returned is nrow(x)*(nrow(x)-1)/2.

Author(s)

Sarah Goslee

See Also

`dist`, `distance`

Examples

```
data(graze)
system.time(graze.bc <- bcdist(graze[, -c(1:2)]))
# equivalent to but much faster than:
system.time(graze.bc2 <- distance(graze[, -c(1:2)], "bray-curtis"))

all.equal(graze.bc, graze.bc2)
```

---

bump *Nine-bump spatial pattern*

Description

A two-dimensional artificial "landscape" illustrating the kind of spatial pattern that might be seen across mountain peaks.

Usage

```
data(bump)
```

Format

The format is: int [1:25, 1:25] 2 2 2 2 2 2 2 2 2 2 ... - attr(*, "dimnames")=List of 2 ..$: chr [1:25] "1" "3" "5" "7" ... ..$: chr [1:25] "V1" "V3" "V5" "V7" ...

Author(s)

Sarah Goslee

See Also

bump.pmgram, pmgram

Examples

```
data(bump)
image(bump)
```

`bump.pmgram` *Nine-bump spatial pattern*

Description

An object of class mgram for use in the example for `pmgram`. Many of the functions in `ecodist` take a long time to run, so prepared examples have been included.

Usage

```
data(bump.pmgram)
```

Format

See `pmgram` for current format specification.

Author(s)

Sarah Goslee

See Also

`bump`, `pmgram`

Examples

```
data(bump)

par(mfrow=c(1, 2))
image(bump, col=gray(seq(0, 1, length=5)))

z <- as.vector(bump)
x <- rep(1:25, times=25)
y <- rep(1:25, each=25)

X <- col(bump)
Y <- row(bump)
# calculate dissimilarities for data and space
geo.dist <- dist(cbind(as.vector(X), as.vector(Y)))
value.dist <- dist(as.vector(bump))

### pgram() is time-consuming, so this was generated
### in advance and saved.
### set.seed(1234)
### bump.pmgram <- pmgram(value.dist, geo.dist, nperm=10000)
### save(bump.pmgram, file="ecodist/data/bump.pmgram.rda")

data(bump.pmgram)
plot(bump.pmgram)
```

`cor2m` *Two-matrix correlation table*

Description

Generate a correlation table between the variables of two data sets, originally for comparing species abundances and environmental variables.

Usage

```
cor2m(x, y, trim = TRUE, alpha = 0.05)
```

Arguments

`x`	A matrix or data frame of environmental (or other) variables matching the sites of x
`y`	A matrix or data frame of species (or other) variables
`trim`	If trim is TRUE, set rho<critical value(alpha) to 0
`alpha`	alpha p-value to use with trim, by default 0.05

Details

cor2m generates a correlation table between the variables of two matrices. The original use case is to compare species abundances and environmental variables. It results in a data frame with species (or the first matrix) as columns and environmental variables (or the second matrix) as rows, so it's easy to scan. Correlations less than a user-specified alpha (0.05 by default) can be set to NA. cor2m generates a correlation table between the variables of two matrices. The original use case is to compare species abundances and environmental variables. The result has species (or the first matrix) as columns and environmental variables (or the second matrix) as rows, so it's easy to scan. Correlations less than a user-specified alpha can be set to NA. If trim, correlations less than the critical value for the provided alpha are set to to NA. The critical value is computed as a t-test with n-2 df. cor2m(x, y, trim=FALSE) is equivalent to cor(x, y)

Value

Returns a data frame of correlations between the variables of 2 data frames.

Author(s)

Dean Urban

Examples

```
data(graze)
speciesdata <- graze[, 3:7]
envdata <- graze[, 1:2]
sppenv.cor <- cor2m(envdata, speciesdata)
print(sppenv.cor, na.print="")
```

`corgen` *Generate correlated data*

Description

Generate correlated data of a given length.

Usage

```
corgen(len, x, r, population = FALSE, epsilon = 0)
```

Arguments

`len`	Length of vectors.
`x`	Independent data. If x is specified, the population parameter is automatically set to TRUE.
`r`	Desired correlation between data vectors.
`population`	TRUE for vectors drawn from two populations with correlation r, otherwise r is the sample correlation.
`epsilon`	Desired tolerance.

Details

Either x or len must be specified. If epsilon = 0, it has no effect, otherwise the sampling process is repeated until the sample correlation is within epsilon of r. This option allows the production of exactly-correlated data, within the limits of epsilon. Setting epsilon > 0 invalidates the population setting; data will be correlated within that range, rather than sampled from that population.If epsilon = 0, it has no effect, otherwise the sampling process is repeated until the sample correlation is within epsilon of r. This option allows the production of exactly-correlated data, within the limits of epsilon. Setting epsilon > 0 invalidates the population setting; data will be correlated within that range, rather than sampled from that population.If epsilon = 0, it has no effect, otherwise the sampling process is repeated until the sample correlation is within epsilon of r. This option allows the production of exactly-correlated data, within the limits of epsilon. Setting epsilon > 0 invalidates the population setting; data will be correlated within that range, rather than sampled from that population.

Value

`x`	First data vector, either generated by corgen or given by the user.
`y`	Second data vector.

Author(s)

Sarah Goslee

Examples

```
# create two random variables of length 100 with correlation
# of 0.10 +/- 0.01
xy <- corgen(len=100, r=.1, epsilon=0.01)
with(xy, cor(x, y))

# create two random variables of length 100 drawn from a population with
# a correlation of -0.82
xy <- corgen(len=100, r=-0.82, population=TRUE)
with(xy, cor(x, y))

# create a variable y within 0.01 of the given correlation to x
x <- 1:100
y <- corgen(x=x, r=.5, epsilon=.01)$y
cor(x, y)
```

---

crosstab *Data formatting*

Description

Converts field data of the form site, species, observation into a site by species data frame.

Usage

```
crosstab(rowlab, collab, values, type = "sum", data, allrows, allcols,
na.as.0 = TRUE, check.names = TRUE, ...)
```

Arguments

`rowlab`	row labels, e.g. site names.
`collab`	column labels, e.g. species names.
`values`	data values.
`data`	optional data frame from which to take rowlab, collab and/or values.
`type`	function to use to combine data, one of "sum" (default), "min", "max", "mean", "count".
`allrows`	optional, list of all desired row names that may not appear in rowlab.
`allcols`	optional, list of all desired column names that may not appear in collab.
`na.as.0`	if TRUE, all NA values are replaced with 0.
`check.names`	if FALSE, data frame names are not checked for syntactic validity, so that they match the input categories. Otherwise make.names() is used to adjust them.
`...`	optional arguments to the function specified in type, such as na.rm=TRUE

Details

Field data are often recorded as a separate row for each site-species combination. This function reformats such data into a data frame for further analysis based on unique row and column labels. The three vectors should all be the same length (including duplicates). The three vectors may also be provided as names of columns in the data frame specified by the data argument.

If allrows or allcols exists, rows and/or columns of zeros are inserted for any elements of allrows/allcols not present in rowlab/collab.

If values is missing the number of occurrences of combinations of rowlab and collab will be returned. Thus, crosstab(rowlab, collab) is equivalent to table(rowlab, collab).

If type is "count", the unique combinations of rowlab, collab and values will be returned.

Value

data frame with rowlab as row headings, collab as columns, and values as the data.

Author(s)

Sarah Goslee

Examples

```
# Make a random example
plotnames <- rep(1:5, each = 6)
speciesnames <- rep(c("A", "B", "C"), 10)
freqdata <- runif(30)

# number of samples of each species and plot
crosstab(plotnames, speciesnames)

# can use the data argument
speciesdata <- data.frame(plots = plotnames, species = speciesnames,
  freq = freqdata, stringsAsFactors=FALSE)

# mean frequency by species and plot
crosstab(plots, species, freq, data=speciesdata, type="mean")

# can specify additional possible row or column levels
crosstab(plots, species, freq, data=speciesdata, type="mean", allcols=LETTERS[1:5])
```

---

distance *Calculate dissimilarity/distance metrics*

Description

This function calculates a variety of dissimilarity or distance metrics. Although it duplicates the functionality of dist() and bcdist(), it is written in such a way that new metrics can easily be added. distance() was written for extensibility and understandability, and is not necessarily an efficient choice for use with large matrices.

Usage

```
distance(x, method = "euclidean", sprange=NULL, spweight=NULL, icov)
```

Arguments

x	matrix or data frame with rows as samples and columns as variables (such as species). Distances will be calculated for each pair of rows.
method	Currently 7 dissimilarity metrics can be calculated: "euclidean", "bray-curtis", "manhattan", "mahalanobis" (squared Mahalanobis distance), "jaccard", "difference", "sorensen", "gower", "modgower10" (modified Gower, base 10), "modgower2" (modified Gower, base 2). Partial matching will work for selecting a method.
sprange	Gower dissimilarities offer the option of dividing by the species range. If sprange=NULL no range is used. If sprange is a vector of length nrow(x) it is used for standardizing the dissimilarities.
spweight	Euclidean, Manhattan, and Gower dissimilarities allow weighting. If spweight=NULL, no weighting is used. If spweight="absence", then W=0 if both species are absent and 1 otherwise, thus deleting joint absences.
icov	Optional covariance matrix; only used if method="mahalanobis" since Mahalanobis distance requires calculating the variance-covariance matrix for the entire dataset. Providing icov directly makes it possible to calculate distances for a subset of the full dataset.

Value

Returns a lower-triangular distance matrix as an object of class "dist".

Author(s)

Sarah Goslee

See Also

dist, bcdist

Examples

```
data(iris)
iris.bc <- distance(iris[, 1:4], "bray-curtis")

# The effect of specifying icov:

# calculate Mahalanobis distance for the full iris dataset
iris.md <- full(distance(iris[, 1:4], "mahal"))
iris.md[1, 2] # Mahalanobis distance between samples 1 and 2

# calculate Mahalanobis for just one species
setosa.md <- full(distance(iris[iris$Species == "setosa", 1:4], "mahal"))
setosa.md[1, 2] # Mahalanobis distance between samples 1 and 2
```

```
# use the covariance matrix for the full dataset to scale for one species
setosa.scaled.md <- full(distance(iris[iris$Species == "setosa", 1:4],
  "mahal", icov=var(iris[,1:4])))
setosa.scaled.md[1, 2] # Mahalanobis distance between samples 1 and 2
```

---

fixdmat *Distance matrix conversion*

Description

Convert a row-order lower-triangular distance matrix to a full symmetric matrix.

Usage

```
fixdmat(v)
```

Arguments

v	lower-triangular distance matrix in row order.

Details

R distance functions such as dist and bcdist return a lower triangular distance matrix in column order. Some other programs return the lower triangular matrix in row order. To use this matrix in R functions, it must be converted from row order to column order.

Value

full symmetric distance matrix.

Author(s)

Sarah Goslee

See Also

lower, full

Examples

```
x.vec <- seq_len(6)
x.vec

# Make an R-style column order symmetric matrix
full(x.vec)

# Extract the lower triangle from a symmetric matrix
```

```
# in column order
lower(full(x.vec))

# Convert to or from a row order symmetric matrix
fixdmat(x.vec)
lower(fixdmat(x.vec))

fixdmat(c(1, 2, 4, 3, 5, 6))
```

`full` *Full symmetric matrix*

Description

Convert a column order distance matrix to a full symmetric matrix.

Usage

```
full(v)
```

Arguments

`v`	lower-triangular column order distance matrix.

Details

Converts a column order lower-triangular distance matrix as written by R functions into a symmetric matrix. Note that `lower()` used on a 1x1 matrix will return the single element, which may not be the correct behavior in all cases, while `full()` used on a single element will return a 2x2 matrix.

Value

full symmetric matrix.

Author(s)

Sarah Goslee

See Also

`lower`, `fixdmat`

Examples

```
# Given a vector:
x.vec <- seq_len(6)
x.vec

# Make an R-style column order symmetric matrix
full(x.vec)

# Extract the lower triangle from a symmetric matrix
# in column order
lower(full(x.vec))

# Convert to or from a row order symmetric matrix
fixdmat(x.vec)
lower(fixdmat(x.vec))

fixdmat(c(1, 2, 4, 3, 5, 6))
```

`graze` *Site information and grazed vegetation data.*

Description

This data frame contains site location, landscape context and dominant plant species abundances for 25 of the plant species found in 50 grazed pastures in the northeastern United States. Elements are the mean values for canopy cover for ten 0.5 x 2 m quadrats.

Usage

```
data(graze)
```

Format

A data frame with 50 observations on the following 25 variables.

`sitelocation` Site location along a geographic gradient.

`forestpct` Percentage forest cover within a 1-km radius.

`ACMI2` Percentage canopy cover.

`ANOD` Percentage canopy cover.

`ASSY` Percentage canopy cover.

`BRIN2` Percentage canopy cover.

`CIAR4` Percentage canopy cover.

`CIIN` Percentage canopy cover.

`CIVU` Percentage canopy cover.

`DAGL` Percentage canopy cover.

`ELRE4` Percentage canopy cover.

`GAMO` Percentage canopy cover.

`LOAR10` Percentage canopy cover.

`LOCO6` Percentage canopy cover.

`LOPE` Percentage canopy cover.

`OXST` Percentage canopy cover.

`PLMA2` Percentage canopy cover.

`POPR` Percentage canopy cover.

`PRVU` Percentage canopy cover.

`RAAC3` Percentage canopy cover.

`RUCR` Percentage canopy cover.

`SORU2` Percentage canopy cover.

`STGR` Percentage canopy cover.

`TAOF` Percentage canopy cover.

`TRPR2` Percentage canopy cover.

`TRRE3` Percentage canopy cover.

`VEOF2` Percentage canopy cover.

Details

Site locations fall along a southwest-northeast transect through the northeastern United States. This is a synthetic gradient calculated from latitude and longitude. Forest landcover is taken from the USGS 1992 National Land Cover Dataset. All forest classes were combined, and the percentage within 1 km of the sample sites was calculated using a GIS.

Author(s)

Sarah Goslee

Source

Details of these data are available in Tracy and Sanderson (2000) and Goslee and Sanderson (2010).
The 1992 NLCD data can be obtained from http://www.mrlc.gov/. Species codes are from http://plants.usda.gov (2010).

References

Tracy, B.F. and M.A. Sanderson. 2000. Patterns of plant species richness in pasture lands of the northeast United States. Plant Ecology 149:169-180.

Goslee, S.C., Sanderson, M.A. 2010. Landscape Context and Plant Community Composition in Grazed Agricultural Systems. Landscape Ecology 25:1029-1039.

Examples

```
data(graze)
```

`iris.fit` *Example of adding to an ordination*

Description

A fitted ordination for use in the example for `addord`. Many of the functions in `ecodist` take a long time to run, so prepared examples have been included.

Usage

```
data(iris.fit)
```

Format

The format of this object is a list with: X1, X2, etc: ordination configuration: coordinates for each point. stress: goodness of fit for each point. isTrain: logical vector indicating whether each point was used in the original ordination.

Author(s)

Sarah Goslee

See Also

`nmds`, `addord`

Examples

```
data(iris)
iris.d <- dist(iris[,1:4])

### nmds() is timeconsuming, so this was generated
### in advance and saved.
### set.seed(1234)
### iris.nmds <- nmds(iris.d, nits=20, mindim=1, maxdim=4)
### save(iris.nmds, file="ecodist/data/iris.nmds.rda")
data(iris.nmds)

# choose the best two-dimensional solution to work with
iris.nmin <- min(iris.nmds, dims=2)

# generate new data points to add to the ordination
# this might be new samples, or a second dataset

iris.new <- structure(list(Sepal.Length = c(4.6, 4.9, 5.4, 5.2, 6, 6.5, 6,
6.8, 7.3), Sepal.Width = c(3.2, 3.5, 3.6, 2.3, 2.8, 3, 2.7, 3.1,
3.2), Petal.Length = c(1.2, 1.5, 1.5, 3.5, 4.1, 4.2, 4.8, 5,
5.7), Petal.Width = c(0.26, 0.26, 0.26, 1.2, 1.3, 1.4, 1.8, 2,
2), Species = structure(c(1L, 1L, 1L, 2L, 2L, 2L, 3L, 3L, 3L), .Label = c("setosa",
```

```
"versicolor", "virginica"), class = "factor")), .Names = c("Sepal.Length",
"Sepal.Width", "Petal.Length", "Petal.Width", "Species"), class = "data.frame",
row.names = c(NA, -9L))

# provide a dist object containing original and new data
# provide a logical vector indicating which samples were used to
# construct the original configuration

iris.full <- rbind(iris, iris.new)
all.d <- dist(iris.full[,1:4])
is.orig <- c(rep(TRUE, nrow(iris)), rep(FALSE, nrow(iris.new)))

### addord() is timeconsuming, so this was generated
### in advance and saved.
### set.seed(1234)
### iris.fit <- addord(iris.nmin, iris.full[,1:4], all.d, is.orig, maxit=100)
### save(iris.fit, file="ecodist/data/iris.fit.rda")
data(iris.fit)

plot(iris.fit$conf, col=iris.full$Species, pch=c(18, 4)[is.orig + 1],
    xlab="NMDS 1", ylab="NMDS 2")
title("Demo: adding points to an ordination")
legend("bottomleft", c("Training set", "Added point"), pch=c(4, 18))
legend("topright", levels(iris$Species), fill=1:3)
```

`iris.nmds` *Example for nmds*

Description

An object of class nmds for use in the example for `nmds`. Many of the functions in `ecodist` take a long time to run, so prepared examples have been included.

Usage

```
data(iris.nmds)
```

Format

See `nmds` for current format specification.

Author(s)

Sarah Goslee

See Also

`nmds`

Examples

```
data(iris)
iris.d <- dist(iris[,1:4])

### nmds() is timeconsuming, so this was generated
### in advance and saved.
### set.seed(1234)
### iris.nmds <- nmds(iris.d, nits=20, mindim=1, maxdim=4)
### save(iris.nmds, file="ecodist/data/iris.nmds.rda")
data(iris.nmds)

# examine fit by number of dimensions
plot(iris.nmds)
```

`iris.vf` *Example for vector fitting on ordination*

Description

An object of class vf for use in the examples for `nmds` and `vf`. Many of the functions in `ecodist` take a long time to run, so prepared examples have been included.

Usage

```
data(iris.vf)
```

Format

See `vf` for current format specification.

Author(s)

Sarah Goslee

See Also

`nmds`, `vf`

Examples

```
data(iris)
iris.d <- dist(iris[,1:4])

### nmds() is timeconsuming, so this was generated
### in advance and saved.
### set.seed(1234)
```

```
### iris.nmds <- nmds(iris.d, nits=20, mindim=1, maxdim=4)
### save(iris.nmds, file="ecodist/data/iris.nmds.rda")
data(iris.nmds)

# examine fit by number of dimensions
plot(iris.nmds)

# choose the best two-dimensional solution to work with
iris.nmin <- min(iris.nmds, dims=2)

# rotate the configuration to maximize variance
iris.rot <- princomp(iris.nmin)$scores

# rotation preserves distance apart in ordination space
cor(dist(iris.nmin), dist(iris.rot))

# fit the data to the ordination as vectors
### vf() is timeconsuming, so this was generated
### in advance and saved.
### set.seed(1234)
### iris.vf <- vf(iris.nmin, iris[,1:4], nperm=1000)
### save(iris.vf, file="ecodist/data/iris.vf.rda")
data(iris.vf)

plot(iris.nmin, col=as.numeric(iris$Species), pch=as.numeric(iris$Species), main="NMDS")
plot(iris.vf)
```

---

iris.vfrot *Example for vector fitting on rotated ordination*

Description

An object of class vf for use in the examples for `nmds` and `vf`. Many of the functions in `ecodist` take a long time to run, so prepared examples have been included.

Usage

```
data(iris.vfrot)
```

Format

See `vf` for current format specification.

Author(s)

Sarah Goslee

See Also

nmds, vf

Examples

```
data(iris)
iris.d <- dist(iris[,1:4])

### nmds() is timeconsuming, so this was generated
### in advance and saved.
### set.seed(1234)
### iris.nmds <- nmds(iris.d, nits=20, mindim=1, maxdim=4)
### save(iris.nmds, file="ecodist/data/iris.nmds.rda")
data(iris.nmds)

# examine fit by number of dimensions
plot(iris.nmds)

# choose the best two-dimensional solution to work with
iris.nmin <- min(iris.nmds, dims=2)

# rotate the configuration to maximize variance
iris.rot <- princomp(iris.nmin)$scores

# rotation preserves distance apart in ordination space
cor(dist(iris.nmin), dist(iris.rot))

# fit the data to the ordination as vectors
### vf() is timeconsuming, so this was generated
### in advance and saved.
### set.seed(1234)
### iris.vf <- vf(iris.nmin, iris[,1:4], nperm=1000)
### save(iris.vf, file="ecodist/data/iris.vf.rda")
data(iris.vf)

# repeat for the rotated ordination
### vf() is timeconsuming, so this was generated
### in advance and saved.
### set.seed(1234)
### iris.vfrot <- vf(iris.rot, iris[,1:4], nperm=1000)
### save(iris.vfrot, file="ecodist/data/iris.vfrot.rda")
data(iris.vfrot)

par(mfrow=c(1,2))
plot(iris.nmin, col=as.numeric(iris$Species), pch=as.numeric(iris$Species), main="NMDS")
plot(iris.vf)
plot(iris.rot, col=as.numeric(iris$Species), pch=as.numeric(iris$Species), main="Rotated NMDS")
plot(iris.vfrot)
```

lower — *Lower-triangular matrix*

Description

Convert a symmetric distance matrix to a column order lower triangular matrix.

Usage

```
lower(m)
```

Arguments

`m`	a symmetric distance matrix.

Details

Converts a symmetric matrix, for example a dissimilarity matrix, into a column order lower-triangular matrix. This may be useful to format the input for certain clustering and ordination functions. Note that `lower()` used on a 1x1 matrix will return the single element, which may not be the correct behavior in all cases, while `full()` used on a single element will return a 2x2 matrix.

Value

column order lower triangular matrix.

Author(s)

Sarah Goslee

See Also

`full`, `fixdmat`

Examples

```
x.vec <- seq_len(6)
x.vec

# Make an R-style column order symmetric matrix
full(x.vec)

# Extract the lower triangle from a symmetric matrix
# in column order
lower(full(x.vec))

# Convert to or from a row order symmetric matrix
fixdmat(x.vec)
lower(fixdmat(x.vec))
```

```
fixdmat(c(1, 2, 4, 3, 5, 6))
```

---

mantel *Mantel test*

Description

Simple and partial Mantel tests, with options for ranked data, permutation tests, and bootstrapped confidence limits.

Usage

```
mantel(formula = formula(data), data = sys.parent(), nperm = 1000,
    mrank = FALSE, nboot = 500, pboot = 0.9, cboot = 0.95)
```

Arguments

`formula`	formula describing the test to be conducted. For this test, y ~ x will perform a simple Mantel test, while y ~ x + z1 + z2 + z3 will do a partial Mantel test of the relationship between x and y given z1, z2, z3. All variables can be either a distance matrix of class dist or vectors of dissimilarities.
`data`	an optional dataframe containing the variables in the model as columns of dissimilarities. By default the variables are taken from the current environment.
`nperm`	number of permutations to use. If set to 0, the permutation test will be omitted.
`mrank`	if this is set to FALSE (the default option), Pearson correlations will be used. If set to TRUE, the Spearman correlation (correlation ranked distances) will be used.
`nboot`	number of iterations to use for the bootstrapped confidence limits. If set to 0, the bootstrapping will be omitted.
`pboot`	the level at which to resample the data for the bootstrapping procedure.
`cboot`	the level of the confidence limits to estimate.

Details

If only one independent variable is given, the simple Mantel r (r12) is calculated. If more than one independent variable is given, the partial Mantel r (ryx|x1 ...) is calculated by permuting one of the original dissimilarity matrices. The bootstrapping is actually resampling without replacement, because duplication of samples is not useful in a dissimilarity context (the dissimilarity of a sample with itself is zero). Resampling within dissimilarity values is inappropriate, just as for permutation.

Value

`mantelr`	Mantel coefficient.
`pval1`	one-tailed p-value (null hypothesis: r <= 0).
`pval2`	one-tailed p-value (null hypothesis: r >= 0).
`pval3`	two-tailed p-value (null hypothesis: r = 0).
`llim`	lower confidence limit.
`ulim`	upper confidence limit.

Author(s)

Sarah Goslee

References

Mantel, N. 1967. The detection of disease clustering and a generalized regression approach. Cancer Research 27:209-220.

Smouse, P.E., J.C. Long and R.R. Sokal. 1986. Multiple regression and correlation extensions of the Mantel test of matrix correspondence. Systematic Zoology 35:62 7-632.

Goslee, S.C. and Urban, D.L. 2007. The ecodist package for dissimilarity-based analysis of ecological data. Journal of Statistical Software 22(7):1-19.

Goslee, S.C. 2010. Correlation analysis of dissimilarity matrices. Plant Ecology 206(2):279-286.

See Also

`mgram`, `mgroup`

Examples

```
data(graze)

grasses <- graze[, colnames(graze) %in% c("DAGL", "LOAR10", "LOPE", "POPR")]
legumes <- graze[, colnames(graze) %in% c("LOCO6", "TRPR2", "TRRE3")]

grasses.bc <- bcdist(grasses)
legumes.bc <- bcdist(legumes)

space.d <- dist(graze$sitelocation)
forest.d <- dist(graze$forestpct)

# Mantel test: is the difference in forest cover between sites
# related to the difference in grass composition between sites?
mantel(grasses.bc ~ forest.d)

# Mantel test: is the geographic distance between sites
# related to the difference in grass composition between sites?
mantel(grasses.bc ~ space.d)
```

```
# Partial Mantel test: is the difference in forest cover between sites
# related to the difference in grass composition once the
# linear effects of geographic distance are removed?
mantel(grasses.bc ~ forest.d + space.d)


# Mantel test: is the difference in forest cover between sites
# related to the difference in legume composition between sites?
mantel(legumes.bc ~ forest.d)

# Mantel test: is the geographic distance between sites
# related to the difference in legume composition between sites?
mantel(legumes.bc ~ space.d)

# Partial Mantel test: is the difference in forest cover between sites
# related to the difference in legume composition once the
# linear effects of geographic distance are removed?
mantel(legumes.bc ~ forest.d + space.d)


# Is there nonlinear pattern in the relationship with geographic distance?
par(mfrow=c(2, 1))
plot(mgram(grasses.bc, space.d, nclass=8))
plot(mgram(legumes.bc, space.d, nclass=8))
```

`mgram` *Mantel correlogram*

Description

Calculates simple Mantel correlograms.

Usage

```
mgram(species.d, space.d, breaks, nclass, stepsize, nperm = 1000,
    mrank = FALSE, nboot = 500, pboot = 0.9, cboot = 0.95,
    alternative = "two.sided", trace = FALSE)
```

Arguments

`species.d`	lower-triangular dissimilarity matrix.
`space.d`	lower-triangular matrix of geographic distances.
`breaks`	locations of class breaks. If specified, overrides nclass and stepsize.
`nclass`	number of distance classes. If not specified, Sturge's rule will be used to determine an appropriate number of classes.
`stepsize`	width of each distance class. If not specified, nclass and the range of space.d will be used to calculate an appropriate default.
`nperm`	number of permutations to use. If set to 0, the permutation test will be omitted.

`mrank` if this is set to FALSE (the default option), Pearson correlations will be used. If set to TRUE, the Spearman correlation (correlation ranked distances) will be used.

`nboot` number of iterations to use for the bootstrapped confidence limits. If set to 0, the bootstrapping will be omitted.

`pboot` the level at which to resample the data for the bootstrapping procedure.

`cboot` the level of the confidence limits to estimate.

`alternative` default is "two.sided", and returns p-values for H0: rM = 0. The alternative is "one.sided", which returns p-values for H0: rM <= 0.

`trace` if TRUE, returns progress indicators.

Details

This function calculates Mantel correlograms. The Mantel correlogram is essentially a multivariate autocorrelation function. The Mantel r represents the dissimilarity in variable composition (often species composition) at a particular lag distance.

Value

Returns an object of class mgram, which is a list with two elements. mgram is a matrix with one row for each distance class and 6 columns:

`lag` midpoint of the distance class.

`ngroup` number of distances in that class.

`mantelr` Mantel r value.

`pval` p-value for the test chosen.

`llim` lower bound of confidence limit for mantelr.

`ulim` upper bound of confidence limit for mantelr.

resids is NA for objects calculated by mgram().

Author(s)

Sarah Goslee

References

Legendre, P. and M. Fortin. 1989. Spatial pattern and ecological analysis. Vegetatio 80:107-138.

See Also

`mantel`, `plot.mgram`, `pmgram`

Examples

```
# generate a simple surface
x <- matrix(1:10, nrow=10, ncol=10, byrow=FALSE)
y <- matrix(1:10, nrow=10, ncol=10, byrow=TRUE)
z <- x + 3*y
image(z)

# analyze the pattern of z across space
space <- cbind(as.vector(x), as.vector(y))
z <- as.vector(z)
space.d <- distance(space, "eucl")
z.d <- distance(z, "eucl")
z.mgram <- mgram(z.d, space.d, nperm=0)
plot(z.mgram)

#

data(graze)
space.d <- dist(graze$sitelocation)
forest.d <- dist(graze$forestpct)

grasses <- graze[, colnames(graze) %in% c("DAGL", "LOAR10", "LOPE", "POPR")]
legumes <- graze[, colnames(graze) %in% c("LOCO6", "TRPR2", "TRRE3")]

grasses.bc <- bcdist(grasses)
legumes.bc <- bcdist(legumes)

# Does the relationship of composition with distance vary for
# grasses and legumes?
par(mfrow=c(2, 1))
plot(mgram(grasses.bc, space.d, nclass=8))
plot(mgram(legumes.bc, space.d, nclass=8))
```

`mgroup` *Mantel test for groups*

Description

Simple and partial Mantel tests, with options for ranked data, permutation tests, and bootstrapped confidence limits.

Usage

```
mgroup(edist, groups, nperm=1000)
```

Arguments

edist	a dist object or lower triangular distance vector.
groups	a vector of group memberships (numeric, character, or factor), or a matrix or data frame with columns describing multiple sets of groups.
nperm	number of permutations to use. If set to 0, the permutation test will be omitted.

Details

mgroup returns the Mantel correlations for group contrast matrices computed from cluster groups across a range of clustering levels.

Value

nclust	Number of groups tested.
mantelr	Mantel coefficient.
pval1	one-tailed p-value (null hypothesis: r <= 0).

Author(s)

Sarah Goslee

References

Legendre, P. and M. Fortin. 1989. Spatial pattern and ecological analysis. Vegetatio 80:107-138.

See Also

mantel

Examples

```
# Using a model matrix to test group membership

data(iris)
iris.d <- dist(iris[,1:4])
mgroup(iris.d, iris[,5])

# clustering-based example

data(graze)
graze.d <- dist(graze[, -c(1:2)])
graze.hclust <- hclust(graze.d)

clust.groups <- data.frame(
k2 = cutree(graze.hclust, k = 2),
k4 = cutree(graze.hclust, k = 4),
k6 = cutree(graze.hclust, k = 6),
k8 = cutree(graze.hclust, k = 8))
```

```
mgroup(graze.d, clust.groups, nperm=1000)
```

---

min.nmds *Find minimum stress configuration*

Description

Finds minimum stress configuration from output of nmds()

Usage

```
## S3 method for class 'nmds'
min(..., na.rm = FALSE, dims = 2)
nmds.min(x, dims = 2)
```

Arguments

`...`	output from nmds()
`x`	output from nmds()
`dims`	desired dimensionality of result. If dims = 0 then the overall minimum stress configuration is chosen. By default, the best two-dimensional configuration is returned.
`na.rm`	Not used; there should be no NA values in a NMDS configuration.

Details

For back-compatibility, the `nmds.min` function remains.

Value

Configuration of minimum stress ordination (dataframe of coordinates). The stress and r^2 for the minimum stress configuration are stored as attributes.

Author(s)

Sarah Goslee

See Also

`nmds`

Examples

```
data(iris)
iris.d <- dist(iris[,1:4])

### nmds() is timeconsuming, so this was generated
### in advance and saved.
### set.seed(1234)
### iris.nmds <- nmds(iris.d, nits=20, mindim=1, maxdim=4)
### save(iris.nmds, file="ecodist/data/iris.nmds.rda")
data(iris.nmds)

# examine fit by number of dimensions
plot(iris.nmds)

# choose the best two-dimensional solution to work with
iris.nmin <- min(iris.nmds, dims=2)
```

MRM *Multiple Regression on distance Matrices*

Description

Multiple regression on distance matrices (MRM) using permutation tests of significance for regression coefficients and R-squared.

Usage

```
MRM(formula = formula(data), data = sys.parent(), nperm = 1000, method = "linear",
mrank = FALSE)
```

Arguments

formula	formula describing the test to be conducted.
data	an optional dataframe containing the variables in the model as columns of dissimilarities. By default the variables are taken from the current environment.
nperm	number of permutations to use. If set to 0, the permutation test will be omitted.
mrank	if this is set to FALSE (the default option), Pearson correlations will be used. If set to TRUE, the Spearman correlation (correlation ranked distances) will be used.
method	if "linear", the default, uses multiple regression analysis. If "logistic", performs logistic regression with appropriate permutation testing. Note that this may be substantially slower.

Details

Performs multiple regression on distance matrices following the methods outlined in Legendre et al. 1994. Specificaly, the permutation test uses a pseudo-t test to assess significance, rather than using the regression coefficients directly.

Value

`coef`	A matrix with regression coefficients and associated p-values from the permutation test (using the pseudo-t of Legendre et al. 1994).
`r.squared`	Regression R-squared and associated p-value from the permutation test (linear only).
`F.test`	F-statistic and p-value for overall F-test for lack of fit (linear only).
`dev`	Residual deviance, degrees of freetom, and associated p-value (logistic only).

Author(s)

Sarah Goslee

References

Lichstein, J. 2007. Multiple regression on distance matrices: A multivariate spatial analysis tool. Plant Ecology 188: 117-131.

Legendre, P.; Lapointe, F. and Casgrain, P. 1994. Modeling brain evolution from behavior: A permutational regression approach. Evolution 48: 1487-1499.

See Also

`mantel`

Examples

```
data(graze)

# This grass is related to forest cover but not location
MRM(dist(LOAR10) ~ dist(sitelocation) + dist(forestpct), data=graze, nperm=100)

# This legume is related to location but not forest cover
MRM(dist(TRRE3) ~ dist(sitelocation) + dist(forestpct), data=graze, nperm=100)
```

nmds *Non-metric multidimensional scaling*

Description

Non-metric multidimensional scaling.

Usage

```
nmds(dmat, mindim = 1, maxdim = 2, nits = 10, iconf, epsilon = 1e-12,
    maxit = 500, trace = FALSE)
```

Arguments

`dmat`	lower-triangular dissimilarity matrix.
`mindim`	optional, minimum number of dimensions to use.
`maxdim`	optional, maximum number of dimensions to use.
`nits`	optional, number of separate ordinations to use.
`iconf`	optional, initial configuration. If not specified, then a random configuration is used.
`epsilon`	optional, acceptable difference in stress.
`maxit`	optional, maximum number of iterations.
`trace`	if TRUE, will write progress indicator to the screen.

Details

The goal of NMDS is to find a configuration in a given number of dimensions which preserves rank-order dissimilarities as closely as possible. The number of dimensions must be specified in advance. Because NMDS is prone to finding local minima, several random starts must be used. Stress is used as the measure of goodness of fit. A lower stress indicates a better match between dissimilarity and ordination. As of ecodist 1.9, the stress calculation used is the same as in `MASS:isoMDS`. In previous versions it was monotonically related, so the same configurations were produced, but the absolute value was different.

Value

`conf`	list of configurations.
`stress`	list of final stress values.
`r2`	total variance explained by each configuration.

The first results are for the lowest number of dimensions (total number is (mindim - maxdim + 1) * nits).

Author(s)

Sarah Goslee

References

Kruskal, J.B. 1964. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika 29:1-27.

Minchin, P.R. 1987. An evaluation of the relative robustness of techniques for ecological ordination. Vegetatio 96:89-108.

See Also

plot.nmds, min.nmds, vf, addord

Examples

```
data(iris)
iris.d <- dist(iris[,1:4])

### nmds() is timeconsuming, so this was generated
### in advance and saved.
### set.seed(1234)
### iris.nmds <- nmds(iris.d, nits=20, mindim=1, maxdim=4)
### save(iris.nmds, file="ecodist/data/iris.nmds.rda")
data(iris.nmds)

# examine fit by number of dimensions
plot(iris.nmds)

# choose the best two-dimensional solution to work with
iris.nmin <- min(iris.nmds, dims=2)

# rotate the configuration to maximize variance
iris.rot <- princomp(iris.nmin)$scores

# rotation preserves distance apart in ordination space
cor(dist(iris.nmin), dist(iris.rot))

# fit the data to the ordination as vectors
### vf() is timeconsuming, so this was generated
### in advance and saved.
### set.seed(1234)
### iris.vf <- vf(iris.nmin, iris[,1:4], nperm=1000)
### save(iris.vf, file="ecodist/data/iris.vf.rda")
data(iris.vf)

# repeat for the rotated ordination
### vf() is timeconsuming, so this was generated
### in advance and saved.
### set.seed(1234)
### iris.vfrot <- vf(iris.rot, iris[,1:4], nperm=1000)
### save(iris.vfrot, file="ecodist/data/iris.vfrot.rda")
data(iris.vfrot)

par(mfrow=c(1,2))
```

```
plot(iris.nmin, col=as.numeric(iris$Species), pch=as.numeric(iris$Species), main="NMDS")
plot(iris.vf)
plot(iris.rot, col=as.numeric(iris$Species), pch=as.numeric(iris$Species),
    main="Rotated NMDS")
plot(iris.vfrot)


# generate new data points to add to the ordination
# this might be new samples, or a second dataset

iris.new <- structure(list(Sepal.Length = c(4.6, 4.9, 5.4, 5.2, 6, 6.5, 6,
6.8, 7.3), Sepal.Width = c(3.2, 3.5, 3.6, 2.3, 2.8, 3, 2.7, 3.1,
3.2), Petal.Length = c(1.2, 1.5, 1.5, 3.5, 4.1, 4.2, 4.8, 5,
5.7), Petal.Width = c(0.26, 0.26, 0.26, 1.2, 1.3, 1.4, 1.8, 2,
2), Species = structure(c(1L, 1L, 1L, 2L, 2L, 2L, 3L, 3L, 3L), .Label = c("setosa",
"versicolor", "virginica"), class = "factor")), .Names = c("Sepal.Length",
"Sepal.Width", "Petal.Length", "Petal.Width", "Species"), class = "data.frame",
row.names = c(NA, -9L))

# provide a dist object containing original and new data
# provide a logical vector indicating which samples were used to
# construct the original configuration

iris.full <- rbind(iris, iris.new)
all.d <- dist(iris.full[,1:4])
is.orig <- c(rep(TRUE, nrow(iris)), rep(FALSE, nrow(iris.new)))

### addord() is timeconsuming, so this was generated
### in advance and saved.
### set.seed(1234)
### iris.fit <- addord(iris.nmin, iris.full[,1:4], all.d, is.orig, maxit=100)
### save(iris.fit, file="ecodist/data/iris.fit.rda")
data(iris.fit)

plot(iris.fit$conf, col=iris.full$Species, pch=c(18, 4)[is.orig + 1],
    xlab="NMDS 1", ylab="NMDS 2")
title("Demo: adding points to an ordination")
legend("bottomleft", c("Training set", "Added point"), pch=c(4, 18))
legend("topright", levels(iris$Species), fill=1:3)
```

---

pco *Principal coordinates analysis*

Description

Principal coordinates analysis (classical scaling).

Usage

```
pco(x, negvals = "zero", dround = 0)
```

Arguments

x	a lower-triangular dissimilarity matrix.
negvals	if = "zero" sets all negative eigenvalues to zero; if = "rm" corrects for negative eigenvalues using method 1 of Legendre and Anderson 1999.
dround	if greater than 0, attempts to correct for round-off error by rounding to that number of places.

Details

PCO (classical scaling, metric multidimensional scaling) is very similar to principal components analysis, but allows the use of any dissimilarity metric.

Value

values	eigenvalue for each component. This is a measure of the variance explained by each dimension.
vectors	eigenvectors. Each column contains the scores for that dimension.

Author(s)

Sarah Goslee

See Also

princomp, nmds

Examples

```
data(iris)
iris.d <- dist(iris[,1:4])
iris.pco <- pco(iris.d)

# scatterplot of the first two dimensions
plot(iris.pco$vectors[,1:2], col=as.numeric(iris$Species),
  pch=as.numeric(iris$Species), main="PCO", xlab="PCO 1", ylab="PCO 2")
```

---

plot.mgram *Plot a Mantel correlogram*

Description

Plot a Mantel correlogram from an object of S3 class mgram, using solid symbols for significant values.

Usage

```
## S3 method for class 'mgram'
plot(x, pval = 0.05, xlab = "Distance", ylab = "Mantel r", ...)
```

Arguments

`x`	an object of class `mgram`
`pval`	cut-off level for statistical significance.
`xlab`	x-axis label.
`ylab`	y-axis label.
`...`	optional, any additional graphics parameters.

Value

draws a plot (graphics device must be active).

Author(s)

Sarah Goslee

See Also

mgram

Examples

```
# generate a simple surface
x <- matrix(1:10, nrow=10, ncol=10, byrow=FALSE)
y <- matrix(1:10, nrow=10, ncol=10, byrow=TRUE)
z <- x + 3*y
image(z)

# analyze the pattern of z across space
space <- cbind(as.vector(x), as.vector(y))
z <- as.vector(z)
space.d <- distance(space, "eucl")
z.d <- distance(z, "eucl")
z.mgram <- mgram(z.d, space.d, nperm=0)
plot(z.mgram)

#

data(graze)
space.d <- dist(graze$sitelocation)
forest.d <- dist(graze$forestpct)

grasses <- graze[, colnames(graze) %in% c("DAGL", "LOAR10", "LOPE", "POPR")]
legumes <- graze[, colnames(graze) %in% c("LOCO6", "TRPR2", "TRRE3")]

grasses.bc <- bcdist(grasses)
legumes.bc <- bcdist(legumes)

# Does the relationship of composition with distance vary for
# grasses and legumes?
```

```
par(mfrow=c(2, 1))
plot(mgram(grasses.bc, space.d, nclass=8))
plot(mgram(legumes.bc, space.d, nclass=8))
```

---

plot.nmds *Plot information about NMDS ordination*

Description

Graphical display of stress and r2 for NMDS ordination along number of dimensions.

Usage

```
## S3 method for class 'nmds'
plot(x, plot = TRUE, xlab = "Dimensions", ...)
```

Arguments

x	an object of S3 class nmds, created by nmds()
plot	optional, if TRUE a figure is produced
xlab	optional, label for x axis of graph
...	optional, other graphics parameters

Value

Produces a two-panel plot with stress and r2 for ordination by number of dimensions. Points show the mean value; lines indicate minimum and maximum.

Author(s)

Dean Urban

See Also

nmds

Examples

```
data(iris)
iris.d <- dist(iris[,1:4])

### nmds() is timeconsuming, so this was generated
### in advance and saved.
### set.seed(1234)
### iris.nmds <- nmds(iris.d, nits=20, mindim=1, maxdim=4)
### save(iris.nmds, file="ecodist/data/iris.nmds.rda")
```

```
data(iris.nmds)

# examine fit by number of dimensions
plot(iris.nmds)

# choose the best two-dimensional solution to work with
iris.nmin <- min(iris.nmds, dims=2)
```

---

plot.vf *Plots fitted vectors onto an ordination diagram*

Description

Add vector fitting arrows to an existing ordination plot.

Usage

```
## S3 method for class 'vf'
plot(x, pval = 1, cex = 0.8, ascale = 0.9, ...)
```

Arguments

`x`	an object of S3 class `vf`, created by `vf()`
`pval`	optional, critical p-value for choosing variables to plot
`cex`	text size
`ascale`	optional, proportion of plot area to use when calculating arrow length
`...`	optional, other graphics parameters

Value

Adds arrows to an existing ordination plot. Only arrows with a p-value less than pval are added. By default, all variables are shown.

Author(s)

Sarah Goslee

See Also

`vf`

Examples

```
# Example of multivariate analysis using built-in iris dataset
data(iris)
iris.d <- dist(iris[,1:4])

### nmds() is timeconsuming, so this was generated
### in advance and saved.
### set.seed(1234)
### iris.nmds <- nmds(iris.d, nits=20, mindim=1, maxdim=4)
### save(iris.nmds, file="ecodist/data/iris.nmds.rda")
data(iris.nmds)

# examine fit by number of dimensions
plot(iris.nmds)

# choose the best two-dimensional solution to work with
iris.nmin <- min(iris.nmds, dims=2)

# fit the data to the ordination as vectors
### vf() is timeconsuming, so this was generated
### in advance and saved.
### set.seed(1234)
### iris.vf <- vf(iris.nmin, iris[,1:4], nperm=1000)
### save(iris.vf, file="ecodist/data/iris.vf.rda")
data(iris.vf)
plot(iris.nmin, col=as.numeric(iris$Species), pch=as.numeric(iris$Species), main="NMDS")
plot(iris.vf)
```

---

pmgram *Partial Mantel correlogram*

Description

This function calculates simple and partial multivariate correlograms.

Usage

```
pmgram(data, space, partial, breaks, nclass, stepsize, resids = FALSE, nperm = 1000)
```

Arguments

`data`	lower-triangular dissimilarity matrix. This can be either an object of class dist (treated as one column) or a matrix or data frame with one or two columns, each of which is an independent lower-triangular dissimilarity in vector form.
`space`	lower-triangular matrix of geographic distances.
`partial`	optional, lower-triangular dissimilarity matrix of ancillary data.

`breaks`	locations of class breaks. If specified, overrides nclass and stepsize.
`nclass`	number of distance classes. If not specified, Sturge's rule will be used to determine an appropriate number of classes.
`stepsize`	width of each distance class. If not specified, nclass and the range of space.d will be used to calculate an appropriate default.
`resids`	if resids=TRUE, will return the residuals for each distance class. Otherwise returns 0.
`nperm`	number of permutations to use. If set to 0, the permutation test will be omitted.

Details

This function does four different analyses: If data has 1 column and partial is missing, calculates a multivariate correlogram for data.

If data has 2 columns and partial is missing, calculates a Mantel cross-correlogram, calculating the Mantel r between the two columns for each distance class separately.

If data has 1 column and partial exists, calculates a partial multivariate correlogram based on residuals of data ~ partial.

If data has 2 columns and partial exists, does a partial Mantel cross-correlogram, calculating partial Mantel r for each distance class separately.

The Iwt statistic used for the multivariate correlograms is not the standard Mantel r. For one variable, using Euclidean distance, this metric converges on the familiar Moran autocorrelation. Like the Moran autocorrelation function, this statistic usually falls between -1 and 1, but is not bounded by those limits. Unlike the Moran function, this correlogram can be used for multivariate data, and can be extended to partial tests.

The comparisons in `vignette("dissimilarity", package="ecodist")` may help.

Value

Returns a object of class mgram, which is a list containing two objects: mgram is a matrix with one row for each distance class and 4 columns:

`lag`	midpoint of the distance class.
`ngroup`	number of distances in that class.
`piecer or Iwt`	Mantel r value or appropriate statistic (see Details).
`pval`	two-sided p-value.

resids is a vector of the residuals (if calculated) and can be accessed with the `residuals()` method.

Author(s)

Sarah Goslee

See Also

`mgram`, `mantel`, `residuals.mgram`, `plot.mgram`

Examples

```
data(bump)

par(mfrow=c(1, 2))
image(bump, col=gray(seq(0, 1, length=5)))

z <- as.vector(bump)
x <- rep(1:25, times=25)
y <- rep(1:25, each=25)

X <- col(bump)
Y <- row(bump)
# calculate dissimilarities for data and space
geo.dist <- dist(cbind(as.vector(X), as.vector(Y)))
value.dist <- dist(as.vector(bump))

### pmgram() is time-consuming, so this was generated
### in advance and saved.
### set.seed(1234)
### bump.pmgram <- pmgram(value.dist, geo.dist, nperm=10000)

data(bump.pmgram)
plot(bump.pmgram)

#### Partial pmgram example

# generate a simple surface
# with complex nonlinear spatial pattern

x <- matrix(1:25, nrow=25, ncol=25, byrow=FALSE)
y <- matrix(1:25, nrow=25, ncol=25, byrow=TRUE)

# create z1 and z2 as functions of x, y
# and scale them to [0, 1]
z1 <- x + 3*y
z2 <- y - cos(x)

z1 <- (z1 - min(z1)) / (max(z1) - min(z1))
z2 <- (z2 - min(z2)) / (max(z2) - min(z2))

z12 <- (z1 + z2*2)/3

# look at patterns

layout(matrix(c(
1, 1, 2, 2,
1, 1, 2, 2,
3, 3, 4, 4,
3, 3, 5, 5), nrow=4, byrow=TRUE))
```

```
image(z1, col=gray(seq(0, 1, length=20)), zlim=c(0,1))
image(z2, col=gray(seq(0, 1, length=20)), zlim=c(0,1))
image(z12, col=gray(seq(0, 1, length=20)), zlim=c(0,1))

# analyze the pattern of z across space
z1 <- as.vector(z1)
z2 <- as.vector(z2)
z12 <- as.vector(z12)
z1.d <- dist(z1)
z2.d <- dist(z2)
z12.d <- dist(z12)

space <- cbind(as.vector(x), as.vector(y))
space.d <- dist(space)

# take partial correlogram without effects of z1
### pmgram() is time-consuming, so this was generated
### in advance and saved.
### set.seed(1234)
### z.no <- pmgram(z12.d, space.d, nperm=1000, resids=FALSE)
### save(z.no, file="ecodist/data/z.no.rda")
data(z.no)
plot(z.no)


# take partial correlogram of z12 given z1
### pmgram() is time-consuming, so this was generated
### in advance and saved.
### set.seed(1234)
### z.z1 <- pmgram(z12.d, space.d, z2.d, nperm=1000, resids=FALSE)
### save(z.z1, file="ecodist/data/z.z1.rda")
data(z.z1)
plot(z.z1)
```

`residuals.mgram` *Residuals of a Mantel correlogram*

Description

Extracts residuals from an S3 object of class `mgram` (only relevant for objects created by `pmgram{}`).

Usage

```
## S3 method for class 'mgram'
residuals(object, ...)
```

Arguments

`object`	an object of class `mgram`
`...`	additional arguments

Value

vector of residuals.

Author(s)

Sarah Goslee

See Also

pmgram, mgram

Examples

```
#### Partial pmgram example

# generate a simple surface
# with complex nonlinear spatial pattern

x <- matrix(1:10, nrow=10, ncol=10, byrow=FALSE)
y <- matrix(1:10, nrow=10, ncol=10, byrow=TRUE)

# create z1 and z2 as functions of x, y
# and scale them to [0, 1]
z1 <- x + 3*y
z2 <- y - cos(x)

z1 <- (z1 - min(z1)) / (max(z1) - min(z1))
z2 <- (z2 - min(z2)) / (max(z2) - min(z2))

z12 <- (z1 + z2*2)/3

# analyze the pattern of z across space
z1 <- as.vector(z1)
z2 <- as.vector(z2)
z12 <- as.vector(z12)
z1.d <- dist(z1)
z2.d <- dist(z2)
z12.d <- dist(z12)

space <- cbind(as.vector(x), as.vector(y))
space.d <- dist(space)

# take partial correlogram of z12 given z1
z.z1 <- pmgram(z12.d, space.d, z2.d, nperm=0, resids=TRUE)
summary(residuals(z.z1))
```

vf *Vector fitting*

Description

Fits ancillary variables to an ordination configuration.

Usage

```
vf(ord, vars, nperm = 100)
```

Arguments

`ord`	matrix containing a 2-dimensional ordination result with axes as columns.
`vars`	matrix with ancillary variables as columns.
`nperm`	number of permutation for the significance test. If nperm = 0, the test will be omitted.

Details

Vector fitting finds the maximum correlation of the individual variables with a configuration of samples in ordination space.

Value

an object of class vf containing matrix with the first 2 columns containing the scores for every variable in each of the 2 dimensions of the ordination space. r is the maximum correlation of the variable with the ordination space, and pval is the result of the permutation test.

Author(s)

Sarah Goslee

References

Jongman, R.H.G., C.J.F. ter Braak and O.F.R. van Tongeren. 1995. Data analysis in community and landscape ecology. Cambridge University Press, New York.

See Also

`plot.vf`

Examples

```
# Example of multivariate analysis using built-in iris dataset
data(iris)
iris.d <- dist(iris[,1:4])

### nmds() is timeconsuming, so this was generated
### in advance and saved.
### set.seed(1234)
### iris.nmds <- nmds(iris.d, nits=20, mindim=1, maxdim=4)
### save(iris.nmds, file="ecodist/data/iris.nmds.rda")
data(iris.nmds)

# examine fit by number of dimensions
plot(iris.nmds)

# choose the best two-dimensional solution to work with
iris.nmin <- min(iris.nmds, dims=2)

# fit the data to the ordination as vectors
### vf() is timeconsuming, so this was generated
### in advance and saved.
### set.seed(1234)
### iris.vf <- vf(iris.nmin, iris[,1:4], nperm=1000)
### save(iris.vf, file="ecodist/data/iris.vf.rda")
data(iris.vf)
plot(iris.nmin, col=as.numeric(iris$Species), pch=as.numeric(iris$Species), main="NMDS")
plot(iris.vf)
```

---

xdistance *Cross-distance between two datasets.*

Description

Pairwise dissimilarity calculation between rows of one dataset and rows of another, for instance across different sampling periods for the same set of sites.

Usage

```
xdistance(x, y, method = "euclidean")
```

Arguments

x	A site by species or other matrix or data frame.
y	A a second site by species dataset, which must have at least the same columns.

`method` This function calls `distance` to do the calculations, and will accept any method used there, currently: "euclidean", "bray-curtis", "manhattan", "mahalanobis" (squared Mahalanobis distance), "jaccard", "difference", "sorensen", "gower", "modgower10" (modified Gower, base 10), "modgower2" (modified Gower, base 2). Partial matching will work for selecting a method.

Details

This function will calculate rowwise dissimilarities between any pair of matrices or data frames with the same number of columns. Note that the cross-dissimilarity functions are for research purposes, and are not well-tested.

Value

A non-symmetric and possibly not square matrix of dissimilarities of class `xdist`, where `result <- xdistance(x, y)` produces a matrix with `result[a, b]` containing the dissimilarity between `x[a, ]` and `y[b, ]`.

Author(s)

Sarah Goslee

See Also

`distance`, `xmantel`, `xmgram`

Examples

```
data(graze)

### EXAMPLE 1: Square matrices

# take two subsets of sites with different dominant grass abundances
# use cut-offs that produce equal numbers of sites
dom1 <- subset(graze, POPR > 50 & DAGL < 20) #  8 sites
dom2 <- subset(graze, POPR < 50 & DAGL > 20) #  8 sites

# first two columns are site info
dom.xd <- xdistance(dom1[, -c(1,2)], dom2[, -c(1,2)], "bray")

# environmental and spatial distances; preserve rownames
forest.xd <- xdistance(dom1[, "forestpct", drop=FALSE],
    dom2[, "forestpct", drop=FALSE])
sitelocation.xd <- xdistance(dom1[, "sitelocation", drop=FALSE],
    dom2[, "sitelocation", drop=FALSE])

# permutes rows and columns of full nonsymmetric matrix
xmantel(dom.xd ~ forest.xd)
xmantel(dom.xd ~ forest.xd + sitelocation.xd)

plot(xmgram(dom.xd, sitelocation.xd))
```

```
### EXAMPLE 2: Non-square matrices

# take two subsets of sites with different dominant grass abundances
# this produces a non-square matrix

dom1 <- subset(graze, POPR > 45 & DAGL < 20) # 13 sites
dom2 <- subset(graze, POPR < 45 & DAGL > 20) #  8 sites

# first two columns are site info
dom.xd <- xdistance(dom1[, -c(1,2)], dom2[, -c(1,2)], "bray")

# environmental and spatial distances; preserve rownames
forest.xd <- xdistance(dom1[, "forestpct", drop=FALSE],
    dom2[, "forestpct", drop=FALSE])
sitelocation.xd <- xdistance(dom1[, "sitelocation", drop=FALSE],
    dom2[, "sitelocation", drop=FALSE])

# permutes rows and columns of full nonsymmetric matrix
xmantel(dom.xd ~ forest.xd, dims=c(13, 8))
xmantel(dom.xd ~ forest.xd + sitelocation.xd, dims=c(13, 8))

plot(xmgram(dom.xd, sitelocation.xd))
```

`xmantel` *Cross-Mantel test*

Description

Simple and partial cross-Mantel tests, with options for ranked data and permutation tests.

Usage

```
xmantel(formula = formula(data), data = sys.parent(), dims = NA,
   nperm = 1000, mrank = FALSE)
```

Arguments

`formula`	formula describing the test to be conducted. For this test, y ~ x will perform a simple Mantel test, while y ~ x + z1 + z2 + z3 will do a partial Mantel test of the relationship between x and y given z1, z2, z3. All variables should be either non-symmetric square cross-dissimilary matrices of class xdist, or vector forms thereof.
`data`	an optional dataframe containing the variables in the model as columns of dissimilarities. By default the variables are taken from the current environment.
`dims`	if the dissimilarity matrices are not square, the dimensions must be provided as `c(nrow, ncol)`
`nperm`	number of permutations to use. If set to 0, the permutation test will be omitted.
`mrank`	if this is set to FALSE (the default option), Pearson correlations will be used. If set to TRUE, the Spearman correlation (correlation ranked distances) will be used.

Details

If only one independent variable is given, the simple Mantel r (r12) is calculated. If more than one independent variable is given, the partial Mantel r (ryx|x1 ...) is calculated by permuting one of the original dissimilarity matrices. Note that the cross-dissimilarity functions are for research purposes, and are not well-tested.

Value

`mantelr`	Mantel coefficient.
`pval1`	one-tailed p-value (null hypothesis: r <= 0).
`pval2`	one-tailed p-value (null hypothesis: r >= 0).
`pval3`	two-tailed p-value (null hypothesis: r = 0).

Author(s)

Sarah Goslee

See Also

`xdistance`, `xmgram`

Examples

```
data(graze)

### EXAMPLE 1: Square matrices

# take two subsets of sites with different dominant grass abundances
# use cut-offs that produce equal numbers of sites
dom1 <- subset(graze, POPR > 50 & DAGL < 20) #  8 sites
dom2 <- subset(graze, POPR < 50 & DAGL > 20) #  8 sites

# first two columns are site info
dom.xd <- xdistance(dom1[, -c(1,2)], dom2[, -c(1,2)], "bray")

# environmental and spatial distances; preserve rownames
forest.xd <- xdistance(dom1[, "forestpct", drop=FALSE],
    dom2[, "forestpct", drop=FALSE])
sitelocation.xd <- xdistance(dom1[, "sitelocation", drop=FALSE],
    dom2[, "sitelocation", drop=FALSE])

# permutes rows and columns of full nonsymmetric matrix
xmantel(dom.xd ~ forest.xd)
xmantel(dom.xd ~ forest.xd + sitelocation.xd)

plot(xmgram(dom.xd, sitelocation.xd))


### EXAMPLE 2: Non-square matrices
```

```
# take two subsets of sites with different dominant grass abundances
# this produces a non-square matrix

dom1 <- subset(graze, POPR > 45 & DAGL < 20) # 13 sites
dom2 <- subset(graze, POPR < 45 & DAGL > 20) #  8 sites

# first two columns are site info
dom.xd <- xdistance(dom1[, -c(1,2)], dom2[, -c(1,2)], "bray")

# environmental and spatial distances; preserve rownames
forest.xd <- xdistance(dom1[, "forestpct", drop=FALSE],
    dom2[, "forestpct", drop=FALSE])
sitelocation.xd <- xdistance(dom1[, "sitelocation", drop=FALSE],
    dom2[, "sitelocation", drop=FALSE])

# permutes rows and columns of full nonsymmetric matrix
xmantel(dom.xd ~ forest.xd, dims=c(13, 8))
xmantel(dom.xd ~ forest.xd + sitelocation.xd, dims=c(13, 8))

plot(xmgram(dom.xd, sitelocation.xd))
```

---

xmgram *Cross-Mantel correlogram*

Description

Calculates simple Mantel correlograms from cross-distance matrices.

Usage

```
xmgram(species.xd, space.xd, breaks, nclass, stepsize, nperm = 1000,
     mrank = FALSE, alternative = "two.sided", trace = FALSE)
```

Arguments

`species.xd`	non-symmetric square cross-distance matrix.
`space.xd`	non-symmetric square matrix of geographic distances.
`breaks`	locations of class breaks. If specified, overrides nclass and stepsize.
`nclass`	number of distance classes. If not specified, Sturge's rule will be used to determine an appropriate number of classes.
`stepsize`	width of each distance class. If not specified, nclass and the range of space.d will be used to calculate an appropriate default.
`nperm`	number of permutations to use. If set to 0, the permutation test will be omitted.
`mrank`	if this is set to FALSE (the default option), Pearson correlations will be used. If set to TRUE, the Spearman correlation (correlation ranked distances) will be used.
`alternative`	default is "two.sided", and returns p-values for H0: rM = 0. The alternative is "one.sided", which returns p-values for H0: rM <= 0.
`trace`	if TRUE, returns progress indicators.

Details

This function calculates cross-Mantel correlograms. The Mantel correlogram is essentially a multivariate autocorrelation function. The Mantel r represents the dissimilarity in variable composition (often species composition) at a particular lag distance. Note that the cross-dissimilarity functions are for research purposes, and are not well-tested.

Value

Returns an object of class mgram, which is a list with two elements. mgram is a matrix with one row for each distance class and 6 columns:

`lag`	midpoint of the distance class.
`ngroup`	number of distances in that class.
`mantelr`	Mantel r value.
`pval`	p-value for the test chosen.

resids is NA for objects calculated by mgram().

Author(s)

Sarah Goslee

References

Legendre, P. and M. Fortin. 1989. Spatial pattern and ecological analysis. Vegetatio 80:107-138.

See Also

`xdistance` `xmantel`, `plot.mgram`

Examples

```
# Need to develop a cross-dissimilarity example
data(graze)

### EXAMPLE 1: Square matrices

# take two subsets of sites with different dominant grass abundances
# use cut-offs that produce equal numbers of sites
dom1 <- subset(graze, POPR > 50 & DAGL < 20) #  8 sites
dom2 <- subset(graze, POPR < 50 & DAGL > 20) #  8 sites

# first two columns are site info
dom.xd <- xdistance(dom1[, -c(1,2)], dom2[, -c(1,2)], "bray")

# environmental and spatial distances; preserve rownames
forest.xd <- xdistance(dom1[, "forestpct", drop=FALSE],
    dom2[, "forestpct", drop=FALSE])
sitelocation.xd <- xdistance(dom1[, "sitelocation", drop=FALSE],
```

```
    dom2[, "sitelocation", drop=FALSE])

# permutes rows and columns of full nonsymmetric matrix
xmantel(dom.xd ~ forest.xd)
xmantel(dom.xd ~ forest.xd + sitelocation.xd)

plot(xmgram(dom.xd, sitelocation.xd))


### EXAMPLE 2: Non-square matrices

# take two subsets of sites with different dominant grass abundances
# this produces a non-square matrix

dom1 <- subset(graze, POPR > 45 & DAGL < 20) # 13 sites
dom2 <- subset(graze, POPR < 45 & DAGL > 20) #  8 sites

# first two columns are site info
dom.xd <- xdistance(dom1[, -c(1,2)], dom2[, -c(1,2)], "bray")

# environmental and spatial distances; preserve rownames
forest.xd <- xdistance(dom1[, "forestpct", drop=FALSE],
    dom2[, "forestpct", drop=FALSE])
sitelocation.xd <- xdistance(dom1[, "sitelocation", drop=FALSE],
    dom2[, "sitelocation", drop=FALSE])

# permutes rows and columns of full nonsymmetric matrix
xmantel(dom.xd ~ forest.xd, dims=c(13, 8))
xmantel(dom.xd ~ forest.xd + sitelocation.xd, dims=c(13, 8))

plot(xmgram(dom.xd, sitelocation.xd))
```

`z.no` *Example for pmgram*

Description

An object of class mgram for use in the example for `pmgram`. Many of the functions in `ecodist` take a long time to run, so prepared examples have been included.

Usage

```
data(z.no)
```

Format

See `pmgram` for current format specification.

Author(s)

Sarah Goslee

See Also

pmgram, z.z1,

Examples

```
#### Partial pmgram example

# generate a simple surface
# with complex nonlinear spatial pattern

x <- matrix(1:25, nrow=25, ncol=25, byrow=FALSE)
y <- matrix(1:25, nrow=25, ncol=25, byrow=TRUE)

# create z1 and z2 as functions of x, y
# and scale them to [0, 1]
z1 <- x + 3*y
z2 <- y - cos(x)

z1 <- (z1 - min(z1)) / (max(z1) - min(z1))
z2 <- (z2 - min(z2)) / (max(z2) - min(z2))

z12 <- (z1 + z2*2)/3

# look at patterns

layout(matrix(c(
1, 1, 2, 2,
1, 1, 2, 2,
3, 3, 4, 4,
3, 3, 5, 5), nrow=4, byrow=TRUE))

image(z1, col=gray(seq(0, 1, length=20)), zlim=c(0,1))
image(z2, col=gray(seq(0, 1, length=20)), zlim=c(0,1))
image(z12, col=gray(seq(0, 1, length=20)), zlim=c(0,1))

# analyze the pattern of z across space
z1 <- as.vector(z1)
z2 <- as.vector(z2)
z12 <- as.vector(z12)
z1.d <- dist(z1)
z2.d <- dist(z2)
z12.d <- dist(z12)

space <- cbind(as.vector(x), as.vector(y))
space.d <- dist(space)

# take partial correlogram without effects of z1
### pgram() is time-consuming, so this was generated
### in advance and saved.
### set.seed(1234)
### z.no <- pmgram(z12.d, space.d, nperm=1000, resids=FALSE)
### save(z.no, file="ecodist/data/z.no.rda")
```

```
plot(z.no)

# take partial correlogram of z12 given z1
### pgram() is time-consuming, so this was generated
### in advance and saved.
### set.seed(1234)
### z.z1 <- pmgram(z12.d, space.d, z2.d, nperm=1000, resids=FALSE)
### save(z.z1, file="ecodist/data/z.z1.rda")
plot(z.z1)
```

---

z.z1 *Example for pmgram*

Description

An object of class mgram for use in the example for pmgram. Many of the functions in ecodist take a long time to run, so prepared examples have been included.

Usage

```
data(z.z1)
```

Format

See pmgram for current format specification.

Author(s)

Sarah Goslee

See Also

pmgram, z.no,

Examples

```
#### Partial pmgram example

# generate a simple surface
# with complex nonlinear spatial pattern

x <- matrix(1:25, nrow=25, ncol=25, byrow=FALSE)
y <- matrix(1:25, nrow=25, ncol=25, byrow=TRUE)

# create z1 and z2 as functions of x, y
# and scale them to [0, 1]
z1 <- x + 3*y
z2 <- y - cos(x)

z1 <- (z1 - min(z1)) / (max(z1) - min(z1))
```

```
z2 <- (z2 - min(z2)) / (max(z2) - min(z2))

z12 <- (z1 + z2*2)/3

# look at patterns

layout(matrix(c(
1, 1, 2, 2,
1, 1, 2, 2,
3, 3, 4, 4,
3, 3, 5, 5), nrow=4, byrow=TRUE))

image(z1, col=gray(seq(0, 1, length=20)), zlim=c(0,1))
image(z2, col=gray(seq(0, 1, length=20)), zlim=c(0,1))
image(z12, col=gray(seq(0, 1, length=20)), zlim=c(0,1))

# analyze the pattern of z across space
z1 <- as.vector(z1)
z2 <- as.vector(z2)
z12 <- as.vector(z12)
z1.d <- dist(z1)
z2.d <- dist(z2)
z12.d <- dist(z12)

space <- cbind(as.vector(x), as.vector(y))
space.d <- dist(space)

# take partial correlogram without effects of z1
### pgram() is time-consuming, so this was generated
### in advance and saved.
### set.seed(1234)
### z.no <- pmgram(z12.d, space.d, nperm=1000, resids=FALSE)
### save(z.no, file="ecodist/data/z.no.rda")
plot(z.no)

# take partial correlogram of z12 given z1
### pgram() is time-consuming, so this was generated
### in advance and saved.
### set.seed(1234)
### z.z1 <- pmgram(z12.d, space.d, z2.d, nperm=1000, resids=FALSE)
### save(z.z1, file="ecodist/data/z.z1.rda")
plot(z.z1)
```

Index

∗Topic **datasets**
- bump, 7
- bump.pmgram, 8
- graze, 16
- iris.fit, 18
- iris.nmds, 19
- iris.vf, 20
- iris.vfrot, 21
- z.no, 52
- z.z1, 54

∗Topic **distribution**
- corgen, 10

∗Topic **hplot**
- plot.mgram, 36
- plot.nmds, 38
- plot.vf, 39

∗Topic **manip**
- crosstab, 11
- fixdmat, 14
- full, 15
- lower, 23

∗Topic **multivariate**
- addord, 3
- bcdist, 6
- cor2m, 9
- crosstab, 11
- distance, 12
- fixdmat, 14
- full, 15
- lower, 23
- mantel, 24
- mgram, 26
- mgroup, 28
- min.nmds, 30
- MRM, 31
- nmds, 33
- pco, 35
- plot.mgram, 36
- plot.nmds, 38
- plot.vf, 39
- pmgram, 40
- residuals.mgram, 43
- vf, 45
- xdistance, 46
- xmantel, 48
- xmgram, 50

∗Topic **package**
- ecodist-package, 2

addord, 3, *18*, *34*

bcdist, 6, *13*
bump, 7, *8*
bump.pmgram, *7*, 8

cor2m, *2*, 9
corgen, 10
crosstab, *2*, 11

dist, *6*, *13*
distance, *6*, 12, *47*

ecodist *(*ecodist-package*)*, 2
ecodist-package, 2

fixdmat, 14, *15*, *23*
full, *14*, 15, *23*

graze, 16

iris.fit, 18
iris.nmds, 19
iris.vf, 20
iris.vfrot, 21

lower, *14*, *15*, 23

mantel, *2*, 24, *27*, *29*, *32*, *41*
mgram, *25*, 26, *37*, *41*, *44*
mgroup, *25*, 28

min.nmds, 30, *34*
MRM, 31

nmds, *2*, *18–22*, *30*, 33, *36*, *38*
nmds.min *(*min.nmds*)*, 30

pco, 35
plot.mgram, *27*, 36, *41*, *51*
plot.nmds, *34*, 38
plot.vf, 39, *45*
pmgram, *7, 8*, *27*, 40, *44*, *52–54*
princomp, *36*

residuals.mgram, *41*, 43

vf, *20–22*, *34*, *39*, 45

xdistance, 46, *49*, *51*
xmantel, *2*, *47*, 48, *51*
xmgram, *47*, *49*, 50

z.no, 52, *54*
z.z1, *53*, 54