Package ‘labdsv’

July 22, 2025

Version 2.1-0

Title Ordination and Multivariate Analysis for Ecology

Author David W. Roberts <droberts@montana.edu>

Maintainer David W. Roberts <droberts@montana.edu>

Depends R (>= 3.0), mgcv

Imports cluster, Rtsne, MASS

Suggests optpart

Enhances vegan

Description A variety of ordination and community analyses
useful in analysis of data sets in community ecology.
Includes many of the common ordination methods, with
graphical routines to facilitate their interpretation,
as well as several novel analyses.

License GPL (>= 2)

NeedsCompilation yes

Repository CRAN

Date/Publication 2023-04-10 09:30:02 UTC

Contents

abundtrans *Species Abundance Data Transformation*

Description

Transforms species abundances according to an arbitrary specified vector

Usage

```
abundtrans(comm,code,value)
```

Arguments

`comm`	the original community data.frame
`code`	a vector containing the set of values appearing in the original data.frame
`value`	a vector containing the set of respective values to substitute

Details

Performs a respective substitution to transform specific values in an initial data.frame to other specified values.

Value

a data.frame of transformed abundance data

Note

Vegetation data are often collected in arbitrary abundance schemes (e.g. Braun-Blanquet, Domin, etc.) which have no direct algebraic transformation (e.g. log). This function transforms coded abundances to arbitrary importance values as specified.

Author(s)

David W. Roberts <`droberts@montana.edu`>

See Also

`decostand`, `wisconsin`

Examples

```
data(bryceveg)
old <- c(0.2,0.5,1.0,2.0,3.0,4.0,5.0,6.0)
new <- c(0.2,0.5,3.0,15.0,37.5,62.5,85.0,97.5)
midpoint <- abundtrans(bryceveg,old,new)
```

abuocc *Abundance/Occurrence Graphical Analysis*

Description

Calculates and plots summary statistics about species occurrences in a data frame

Usage

```
abuocc(comm,minabu=0,panel='all')
```

Arguments

`comm`	a community data.frame with samples as rows and species as columns
`minabu`	a minimum abundance threshold species must exceed to be included in the calculations (default=0)
`panel`	controls which of four graphs is drawn, and can be 'all' or integers 1-4

Details

This functions calculates and plots four data summaries about the occurrence of species:

Plots:

1) the number of samples each species occurs in on a log scale, sorted from maximum to minimum

2) the number of species in each sample plot (species richness) from highest to lowest

3) the mean abundance of non-zero values (on a log scale) as a function of the number of plots a species occurs in

4) the total abundance/sample as a function of the plot-level species richness

The third plot allows you to identify individual species with the mouse; the fourth plot allows you to identify individual sample units with the mouse.

Value

Returns an (invisible) list composed of:

`spc.plt`	number of species/sample
`plt.spc`	number of samples each species occurs in
`mean`	mean abundance of each species when present (excluding values smaller than minabu)

Note

It's common in niche theory analyses to calculate the rank abundances of taxa in a sample. This function is similar, but works on multiple samples simultaneously. The spc.plt vector in the returned list can be used anywhere species richness is desired. The plt.spc vector in the returned list can be used to mask out rare species in calculations of sample similarity using `dsvdis` among other purposes.

Author(s)

David W. Roberts <droberts@montana.edu>

See Also

`fisherfit`, `prestonfit`, `radfit`

Examples

```
data(bryceveg) # produces a data.frame called bryceveg
abuocc(bryceveg)
```

`as.dsvord` *Convert existing and external ordinations to dsv format*

Description

This function updates ordinations from previous versions of labdsv and converts ordinations of class 'boral' from package boral, list output objects from package Rtsne, class 'metaMDS' objects from package vegan, or class 'ordiplot' objects from package vegan into objects of class 'dsvord' for plotting and comparison.

Usage

```
as.dsvord(obj)
```

Arguments

`obj`	an object of class nmds, pco, pca, boral, metaMDS, or ordiplot or an output list object from Rtsne

Details

as.dsvord calls internal format-specific conversion functions to produce an object of class 'dsvord' from the given input.

Value

an object of class 'dsvord', i.e. a list with items 'points' and 'type' (optionally more), and attributes 'call' and 'timestamp' and 'class'.

Note

LabDSV recently converted all ordination objects to a single class with an ancillary 'type' specification to differentiate ordination types.

Author(s)

David W. Roberts <droberts@montana.edu>

Examples

```
## Not run: data(bryceveg)
dis.bc <- dsvdis(bryceveg,'bray')
library(vegan)
demo.metaMDS <- metaMDS(bryceveg)
metamds.dsv <- as.dsvord(demo.metaMDS)
demo.ordi <- plot(demo.metaMDS)
ordip.dsv <- as.dsvord(demo.ordi)
library(boral)
demo.boral <- boral(bryceveg,row.eff='random')
boral.dsv <- as.dsvord(demo.boral)

## End(Not run)
```

`brycesite` *Site Data for Bryce Canyon National Park*

Description

Environmental variables recorded at or calculated for each of 160 sample plots in Bryce Canyon National Park, Utah, U.S.A.

Usage

```
data(brycesite)
```

Format

a data.frame with sample units as rows and site variables as columns. Variables are:

plotcode = original plot codes

annrad = annual direct solar radiation in Langleys

asp = slope aspect in degrees

av = aspect value = (1+cosd(asp-30))/2

depth = soil depth = "deep" or "shallow"

east = UTM easting in meters

elev = elevation in feet

grorad = growing season radiation in Langleys

north = UTM northing in meters

pos = topographic position

quad = USGS 7.5 minute quad sheet

slope = percent slope

`bryceveg` *Bryce Canyon Vegetation Data*

Description

Estimates of cover class for all non-tree vascular plant species in 160 $375m^2$ circular sample plots. Species codes are first three letters of genus + first three letters of specific epithet.

Usage

```
data(bryceveg)
```

Format

a data.frame of 160 sample units (rows) and 169 species (columns). Cover is estimated in codes as follows:

0.2	present in the stand but not the plot
0.5	0-1%
1.0	1-5%
2.0	5-25%
3.0	25-50%
4.0	50-75%
5.0	75-95%
6.0	95-100%

`calibrate` *Calculate fitted environmental attributes in an ordination*

Description

Fits a Generalized Additive Model (GAM) for each environmental variable in a data.frame against an ordination.

Usage

```
## S3 method for class 'dsvord'
calibrate(ord,site,dims=1:ncol(ord$points),
          family='gaussian',gamma=1,keep.models=FALSE,...)
```

Arguments

`ord`	an ordination object of class dsvord
`site`	a matrix or data.frame with sample units as rows and environmental variables as columns
`dims`	the specific dimensions of the ordination to consider
`family`	the error distribution specifier for the GAM function
`gamma`	the gamma parameter to control fitting GAM models
`keep.models`	a switch to control saving the individual GAM models
`...`	arguments to pass

Details

The calibrate function sequentially and independently fits a GAM model for each environmental variable as a function of ordination coordinates, using the family and gamma specifiers supplied in the function call, or their defaults. The model fits two or three dimensional models; if the length of dims is greater than three the dimensions are truncated to the first three chosen.

Value

A list object with vector elements aic, dev.expl, adj.rsq, and fitted value matrix. Optionally, if keep.models is TRUE, a list with all of the GAM models fitted. List element aic gives the model AICs for each variable, dev.expl gives the deviance explained, adj.rsq gives the adjusted r-Squared, and fitted gives the expected value of each variable in each sample unit.

Author(s)

David W. Roberts <`droberts@montana.edu`>

See Also

predict for the complementary function that fits GAM models for species

Examples

```
data(bryceveg)
dis.man <- dist(bryceveg,method="manhattan")
demo.nmds <- nmds(dis.man,k=4)
## Not run: res <- calibrate(demo.nmds,brycesite[,c(2,4,7,12)],minocc=10)
```

`ccm` *Community Composition Modeling*

Description

Compares the composition of modeled communities to real data using Bray-Curtis similarity

Usage

```
ccm(model,data)
```

Arguments

`model`	fitted data from a predictive model
`data`	actual data from the modeled communities

Details

The algorithm sweeps through the fitted values and data one sample unit at time calculating the similarity to the simulated community to the real community. The calculation is similarity, not dissimilarity, and results in a vector of length equal to the number of sample units.

The diverse matrix has the diversity of the data in the first column, and the diversity of the simulated or fitted data in the second column.

Value

A list object with two components:

`sim`	a vector of similarities of modeled communities to actual data
`diverse`	Shannon-Weaver diversity values for modeled and real data

Author(s)

David W. Roberts `<droberts@montana.edu>`

Examples

```
data(bryceveg)
bryceveg <- dropspc(bryceveg,4)
bryce.bc <- dsvdis(bryceveg,'bray')
bryce.nmds <- nmds(bryce.bc)
## Not run: bryce.preds <- predict(bryce.nmds,bryceveg)
## Not run: bryce.ccm <- ccm(bryceveg,bryce.preds$fitted)
## Not run: summary(bryce.ccm$sim)
## Not run: boxplot(bryce.ccm$diverse)
```

compspec — *Compositional Specificity Analysis*

Description

Calculates the mean similarity of all plots in which each species occurs

Usage

```
compspec(comm, dis, numitr=100, drop=FALSE, progress=FALSE)
## S3 method for class 'compspec'
plot(x,spc=NULL,pch=1,type='p',col=1,...)
```

Arguments

`comm`	a data frame of community samples, samples as rows, species as columns
`dis`	an object of class ‘dist’ from `dist`, `dsvdis` or `vegdist`
`numitr`	the number of iterations to use to establish the quantiles of the distribution
`drop`	a switch to determine whether to drop species out when calculating their compspec value
`progress`	a switch to control printing out a progress bar
`x`	an object of class compspec
`spc`	an integer code to specify exactly which species drop-out to plot
`pch`	which glyph to plot for species
`type`	which type of plot
`col`	an integer or integer vector) to color the points
`...`	additional arguments to the plot function

Value

a list with several data.frames: ‘vals’ with species name, mean similarity, number of occurrences, and probability of observing as high a mean similarity as observed, and ‘quantiles’ with the distribution of the quantiles of mean similarity for given numbers of occurrences. If drop=TRUE, results specific to dropping out each species in turn are added to the list by species name.

Note

One measure of the habitat specificity of a species is the degree to which a species only occurs in communities that are similar to each other. This function calculates the mean similarity of all samples in which each species occurs, and compares that value to the distribution of mean similarities for randomly generated sets of the same size. The mean similarity of species which only occur once is set to 0, rather than NA.

If drop=TRUE each species is deleted in turn and a new dissimilarity matrix minus that species is calculated for the analysis. This eliminates the bias that part of the similarity of communities being analyzed is due to the known joint occurrence of the species being analyzed.

Author(s)

David W. Roberts <droberts@montana.edu>

See Also

`indval`,`isamic`

Examples

```
data(bryceveg) # returns a vegetation data.frame
dis.bc <- dsvdis(bryceveg,'bray/curtis')
    # returns a Bray/Curtis dissimilarity matrix
compspec(bryceveg,dis.bc)
```

---

concov *Constancy-Coverage Table for Ecological Community Data*

Description

Produces a table of combined species constancy and importance

Usage

```
concov(comm,clustering,digits=1,width=5,typical=TRUE,thresh=10)
```

Arguments

`comm`	a community data.frame, samples as rows and species as columns
`clustering`	(1) an object of class 'clustering', class 'partana', or class 'partition', (2) a vector of integer cluster memberships, (3) a factor vector, or (4) a character vector
`digits`	the number of digits for the importance value of species
`width`	controls the formatting of columns
`typical`	an argument passed to `importance` to control how mean abundance is calculated
`thresh`	a threshold parameter to control the suppression of small details in the output. Species must have >= thresh constancy in at least one type to appear in the output table

Details

concov calls `const` and `importance` and then combines the output in a single table.

Value

a data.frame with factors (combined constancy and coverage) as columns

Note

Constancy-coverage tables are an informative and concise representation of species in classified types. The output format [constancy(mean cover)] follows the convention of the US Forest Service vegetation classifications.

Author(s)

David W. Roberts <droberts@montana.edu>

See Also

const, importance

Examples

```
data(bryceveg)  # returns a vegetation data.frame
data(brycesite) # returns a site data.frame
## Not run: concov(bryceveg,brycesite$quad) # calculates the constancy
                                          # and coverage by USGS quad
## End(Not run)
```

`const` *Constancy Table*

Description

For a classified set of vegetation samples, lists for each species the fraction of samples in each class the species occurs in.

Usage

```
const(comm, clustering, minval = 0, show = minval, digits = 2,
               sort = FALSE, spcord = NULL)
```

Arguments

`comm`	a data.frame of species abundances with samples as rows and species as columns
`clustering`	(1) an object of class 'clustering', class 'partana', or class 'partition', (2) a vector of numeric cluster memberships, (3) a factor vector, or (4) a character vector.
`minval`	the minimum constancy a species must have in at least one class to be included in the output
`show`	the minimum constancy a species must have to show a printed value
`digits`	the number of digits to report in the table
`sort`	a switch to control interactive re-ordering of the output table
`spcord`	a vector of integers to specify the order in which species should be listed in the table

Details

Produces a table with species as rows, and species constancy in clusters as columns.

The 'clustering' vector represents a classification of the samples that the table summarizes. It may result from a cluster analysis, partitioning an ordination, subjective partitioning of a vegetation table, or other source.

The 'minval' argument is used to emphasize the dominant species and suppress the rare species. Vegetation tables are often very sparse, and this argument simplifies making them more compact.

The 'digits' argument limits the reported precision of the calculations. Generally, relatively low precision is adequate and perhaps more realistic.

The 'spcord' argument specifies the order species are listed in a table. You can use the reverse of the number of occurrences to get dominant species at the top to rarer at the bottom, use fidelity values for the ordered clusters, or possibly the order of species centroids in an ordination.

Value

a data.frame with species as rows, classes as columns, with fraction of occurrence of species in classes.

Note

Constancy tables are often used in vegetation classification to calculate or present characteristic species for specific classes or types. 'const' may be combined with 'importance' and 'vegtab' to achieve a vegetation table-oriented analysis.

Author(s)

David W. Roberts `<droberts@montana.edu>`

See Also

`importance`, `vegtab`, `vegemite`

Examples

```
data(bryceveg) # returns a data.frame called bryceveg
data(brycesite)
class <- cut(brycesite$elev,10,labels=FALSE)
const(bryceveg,class,minval=0.25)
```

convex *Convex Data Transformation*

Description

Calculates a convex data transformation for a given number of desired classes.

Usage

```
convex(n,b=2,stand=FALSE)
```

Arguments

`n`	the desired number of values
`b`	the base of the exponential function
`stand`	a switch to control standardizing values to a maximum of 1.0

Details

Calculates a series of values where the difference between adjacent values is 1/b the previous difference. With the default b=2 you get an octave scale.

Value

a vector of numeric values

Author(s)

David W. Roberts `<droberts@montana.edu>`

See Also

spcmax, samptot, abundtrans, hellinger

Examples

```
convex(5,2)
```

`defactorize` *Change Factors in Data.frames to Character Vectors*

Description

Looks at each column in a data.frame, and converts factors to character vectors.

Usage

```
defactorize(df)
```

Arguments

`df`	a data.frame

Details

The function simply scans each column in a data.frame looking for factor columns. For each factor column it calls the 'as.character()' function to convert the column to a character vector.

Value

Returns a data.frame where every factor column has been converted to a character vector.

Note

This function simplifies editing data.frames by allowing users to edit character columns (which have no levels constraints) and then converting the results to factors for modeling. It is often used in a cycle of

defactorize(df)

edit the columns as necessary to correct errors or simplify

factorize(df)

Author(s)

David W. Roberts `<droberts@montana.edu>`

See Also

factorize

Examples

```
data(brycesite)
brycesite <- defactorize(brycesite)
brycesite$quad[brycesite$quad=='bp'] <- 'BP'
brycesite <- factorize(brycesite)
```

dematrify *Create Three Column Database Form Data Frame from Sparse Data Frames*

Description

Takes a sparse matrix data frame (typical of ecological abundance data) and converts it into three column database format.

Usage

```
dematrify(comm, filename, sep = ",", thresh = 0)
```

Arguments

`comm`	a sparse data.frame or matrix, with samples as rows and comm as columns
`filename`	the name of the filename to produce
`sep`	the separator to use in separating columns
`thresh`	the minimum abundance to be included in the output

Details

The routine is pure R code to convert data from sparse matrix form to three column database form for export or reduced storage

Value

a data.frame with the first column the sample ID, the second column the taxon ID, and the third column the abundance.

Note

Typically, large ecological data sets are characterized by sparse matrices of taxon abundance in samples with many zeros in the matrix. Because these datasets may be many columns wide, they are difficult to work with in text editors or spreadsheets, and require excessive amount of space for storage. The reduced three column form is suitable for input to databases, and more easily edited.

Author(s)

David W. Roberts `<droberts@montana.edu>`

See Also

`matrify`

Examples

```
library(labdsv)
data(bryceveg)
x <- dematrify(bryceveg)
```

`dga` *Direct Gradient Analysis*

Description

Direct gradient analysis is a graphical representation of the abundance distribution of (typically) species along opposing environmental gradients

Usage

```
dga(z,x,y,step=50,pres="+",abs="-",labcex=1,
    xlab = deparse(substitute(x)), ylab = deparse(substitute(y)),
    pch = 1, title = "", ...)
```

Arguments

`z`	the variable (typically a species abundance) to be plotted
`x`	the variable to use as the x axis
`y`	the variable to use as the y axis
`step`	controls the grid density fed to the GAM surface fitter
`pres`	the symbol to print when a species is present (presence/absence mode)
`abs`	the symbol to print when a species is absent (presence/absence mode)
`labcex`	the character size for contour labels
`xlab`	the x axis legend
`ylab`	the y axis legend
`pch`	the symbol to print in continuous abundance plots
`title`	the title to print
`...`	miscellaneous arguments to pass to par

Details

'dga' interpolates a grid of x,y values from the supplied data and fits a GAM (from `mgcv`) of the z variable to the grid. For presence/absence data (enterd as a logical) it employs a binomial family, for species abundances a negative binomial is employed. The GAM surface is then represented by a contour map and abundance symbols as described above.

Value

a graph of the distribution of the z variable on a grid of x and y is displayed on the current active device.

Note

Direct gradient analysis was promoted by Robert Whittaker and followers as a preferred method of vegetation analysis.

Author(s)

David W. Roberts <droberts@montana.edu>

See Also

gam

Examples

```
data(bryceveg) # returns a data.frame called bryceveg
x <- c(0.2,0.5,1.0,2.0,3.0,4.0,5.0,6.0)
y <- c(0.2,0.5,3.0,15.0,37.5,62.5,85.0,97.5)
cover <- abundtrans(bryceveg,x,y)
data(brycesite)
dga(round(cover$arcpat),brycesite$elev,brycesite$av)
```

---

disana *Dissimilarity Analysis*

Description

Dissimilarity analysis is a graphical analysis of the distribution of values in a dissimilarity matrix

Usage

```
disana(x, panel='all')
```

Arguments

x	an object of class 'dist' such as returned by dist, dsvdis. or vegdist
panel	a switch to specify which panel of graphics should be displayed. Can be either an integer from 1 to 3, or the word 'all'.

Details

Calculates three vectors: the minimum, mean, and maximum dissimilarity for each sample in a dissimilarity matrix. By default it produces three plots: the sorted dissimilarity values, the sorted min, mean, and maximum dissimilarity for each sample, and the mean dissimilarity versus the minimum dissimilarity for each sample. Optionally, you can identify sample plots in the last panel with the mouse.

Value

Plots three graphs to the current graphical device, and returns an (invisible) list with four components:

`min`	the minimum dissimilarity of each sample to all others
`mean`	the mean dissimilarity of each sample to all others
`max`	the maximum dissimilarity of each sample to all others
`plots`	a vector of samples identified in the last panel

Note

Dissimilarity matrices are often large, and difficult to visualize directly. ‘disana’ is designed to highlight aspects of interest in these large matrices. If the first panel shows a long limb of constant maximum value, you should consider recalculating the dissimilarity with a step-across adjustment. The third panel is useful for identifying outliers, which are plots more than 0.5 dissimilar to their nearest neighbor.

Author(s)

David W. Roberts `<droberts@montana.edu>`

Examples

```
data(bryceveg) # returns a data.frame called veg
dis.bc <- dsvdis(bryceveg,'bray/curtis')
disana(dis.bc)
```

`dropplt` *Dropping Plots with Missing Values From Taxon and Site Data Frames*

Description

Looks for plots which have missing values in site or environment data, and deletes those plots from both the community and site data frames.

Usage

```
dropplt(comm,site,which=NULL)
```

Arguments

`comm`	a community data frame with samples as rows and species as columns
`site`	a site or environment data frame with samples as rows and variables as columns
`which`	a switch to specify specific plots to drop from both data.frames

Details

First looks to see that the row names of the community data frame and the site or environment data frame are identical. If not, it prints an error message and exits. If which is NULL, it then looks at the site or environment data frame for plots or samples that have missing values, and deletes those plots from both the community and site data frames. Alternatively, if which is a numeric scalar or vector it deletes the specified plots from both the community and site data.frames.

Value

produces a list with two components:

`site`	the new site data frame

Note

This is a VERY heavy-handed approach to managing missing values. Most R routines (including most of the labdsv package functions) have ways of handling missing values that are fairly graceful. This function simply maintains the correspondence between the community and site data frames while eliminating ALL missing values, and all plots that have missing values.

Author(s)

David W. Roberts `<droberts@montana.edu>`

Examples

```
data(bryceveg)  # returns a data frame called bryceveg
data(brycesite) # returns a data frame called brycesite
demo <- dropplt(bryceveg,brycesite)
newcomm <- demo$comm
newsite <- demo$site
```

`dropspc` *Dropping Species with Few Occurrences*

Description

Eliminates species from the community data frame that occur fewer than or equal to a threshold number of occurrences.

Usage

```
dropspc(comm,minocc=0,minabu=0)
```

Arguments

`comm`	a community data frame
`minocc`	the threshold number of occurrences to be dropped
`minabu`	the threshold minimum abundance to be dropped

Details

The function is useful for eliminating species (columns) from community data frames which never occur, which often happens if you eliminate plots, and those plots are the only ones that contain that species. In addition, many species are rare in data frames, and some algorithms (especially dissimilarity functions and table sorting routines) benefit from smaller, simpler data frames.

Value

Produces a new community data frame

Note

This is a heavy-handed approach to managing rare species in data.frames. It is often possible to write a mask (logical vector) that suppresses the influence of rare species and keeps the original data.frame intact, but this function simplifies data management for some purposes.

Author(s)

David W. Roberts <`droberts@montana.edu`>

Examples

```
data(bryceveg) # returns a data frame called bryceveg
newveg <- dropspc(bryceveg,5) # deletes species which
                              # occur 5 or fewer times
```

`dsvdis` *Dissimilarity Indices and Distance Measures*

Description

This function provides a set of alternative dissimilarity indices and distance metrics for classification and ordination, including weighting by species (columns) and shortest-path adjustment for dissimilarity indices.

Usage

```
dsvdis(x,index,weight=rep(1,ncol(x)),step=0.0,
       diag=FALSE, upper=FALSE)
```

Arguments

`x`	a matrix of observations, samples as rows and variables as columns
`index`	a specific dissimilarity or distance index (see details below)
`weight`	a vector of weights for species (columns)
`step`	a threshold dissimilarity to initiate shortest-path adjustment (0.0 is a flag for no adjustment)
`diag`	a switch to control returning the diagonal (default=FALSE)
`upper`	a switch to control returning the upper (TRUE) or lower (FALSE) triangle

Details

The function calculates dissimilarity or distance between rows of a matrix of observations according to a specific index. Three indices convert the data to presence/absence automatically. In contingency table notation, they are:

steinhaus	$1 - a/(a + b + c)$
sorensen	$1 - 2a/(2a + b + c)$
ochiai	$1 - a/\sqrt{(a + b) * (a + c)}$

Others are quantitative. For variable i in samples x and y:

ruzicka	$1 - \sum \min(x_i, y_i)/\sum \max(x_i, y_i)$
bray/curtis	$1 - \sum[2 * \min(x_i, y_i)]/\sum x_i + y_i$
roberts	$1 - [(x_i + y_i) * \min(x_i, y_i)/\max(x_i, y_i)]/(x_i + y_i)$
chisq	$(exp - obs)/\sqrt{exp}$

The weight argument allows the assignment of weights to individual species in the calculation of plot-to-plot similarity. The weights can be assigned by life-form, indicator value, or for other investigator specific reasons. For the presence/absence indices the weights should be integers; for the quantitative indices the weights should be in the interval [0,1]. The default (rep(1,ncol(x)) is to set all species = 1.

The threshold dissimilarity 'step' sets all values greater than or equal to "step" to 9999.9 and then solves for the shortest path distance connecting plots to other non-9999.9 values in the matrix. Step = 0.0 (the default) is a flag for "no shortest-path correction".

Value

Returns an object of class "dist", equivalent to that from `dist`.

Note

Ecologists have spent a great deal of time and effort examining the properties of different dissimilarity indices and distances for ecological data. Many of these indices should have more general application however. Dissimilarity indices are bounded [0,1], so that samples with no attributes in common cannot be more dissimilar than 1.0, regardless of their separation along hypothetical or real gradients. The shortest-path adjustment provides a partial solution. Pairs of samples more dissimilar than a specified threshold are set to 9999.9, and the algorithm solves for their actual dissimilarity from the transitive closure of the triangle inequality. Essentially, the dissimilarity is replaced by the shortest connected path between points less than the threshold apart. In this way it is possible to obtain dissimilarities much greater than 1.0.

The chi-square distance is not usually employed directly in cluster analysis or ordination, but is provided so that you can calculate correspondence analysis as a principal coordinates analysis (using `cmdscale`) from a simple distance matrix.

Author(s)

David W. Roberts <droberts@montana.edu>

See Also

`dist`, `vegdist`

Examples

```
data(bryceveg)   # returns a data.frame called "bryceveg"
dis.ochiai <- dsvdis(bryceveg,index="ochiai")
dis.bc <- dsvdis(bryceveg,index="bray/curtis")
```

---

dsvls *LabDSV Object ls() Command*

Description

The function searches through all the objects in the specified environment, and determines which ones have specific meaning in LabDSV. It then produces an output of a summary of every known LabDSV object sorted by type.

Usage

```
dsvls(frame=NULL,opt='full')
```

Arguments

`frame`	an environment; if null substitutes parent.frame()
`opt`	a switch for 'full' or 'brief' output

Value

Prints output to the console

Note

It's common that after a while the number of objects in your workspace can get large, and even with disciplined naming of objects the list can get overwhelming. dsvls() attempts to organize and report on the objects LabDSV understands.

Author(s)

David W. Roberts <droberts@montana.edu>

Examples

```
data(bryceveg)
dis.bc <- dsvdis(bryceveg,'bray')
nmds.bc <- nmds(dis.bc,2)
dsvls()
```

---

envrtest *Environmental Distribution Test*

Description

Calculates whether the value of a specified environmental variable has an improbable distribution with respect to a specified vector

Usage

```
envrtest(set,env,numitr=1000,minval=0,replace=FALSE,
     plotit = TRUE, main = paste(deparse(substitute(set)),
     " on ", deparse(substitute(env))))
```

Arguments

`set`	a vector of logical or quantitative values
`env`	the quantitative variable whose distribution is to be tested
`numitr`	the number of randomizations to iterate to calculate probabilities
`minval`	the threshold to use to partition the data into a logical if set is quantitative
`replace`	whether to permute (replace=FALSE) or bootstrap (replace=TRUE) the values in the permutation test
`plotit`	logical; plot results if TRUE
`main`	title for plot if plotted

Details

Calculates the maximum within-set difference in the values of vector ‘env’, and the distribution of the permuted random within-set differences. It then plots the observed difference as a red line, and the sorted permuted differences as a black line and prints the probability of getting such a limited distribution. The probability is calculated by permuting numitr-1 times, counting the number of times the permuted maximum difference is as small or smaller than observed (n), and calculating (n+1)/numitr. To get three-digit probabilities, set numitr=1000 (the default)

Value

Produces a plot on the current graphics device, and an invisible list with the components observed within-set difference and the p-value.

Note

The plot is based on the concept of constraint, or limiting value, and checks to see whether the distribution of a particular variable within a cluster is constrained in an improbable way.

Author(s)

David W. Roberts `<droberts@montana.edu>`

Examples

```
data(bryceveg) # returns a vegetation data.frame
data(brycesite) # returns and environmental data.frame
envrtest(bryceveg$berrep>0,brycesite$elev)
```

`euclidify` *Nearest Euclidean Space Representation of a Dissimilarity Object*

Description

Calculates the nearest Euclidean space representation of a dissimilarity object by iterating the transitive closure of the triangle inequality

Usage

```
euclidify(dis,upper=FALSE,diag=FALSE)
as.euclidean(dis,upper=FALSE,diag=FALSE)
```

Arguments

`dis`	a distance or dissimilarity object returned from `dist`, `vegdist`, or `dsvdis`
`upper`	a logical switch to control whether to return the lower triangle (upper=FALSE) or upper triangle (upper=TRUE) of the distance matrix
`diag`	a logical switch to control whether to return the diagonal of the distance matrix

Details

Implements a constrained iteration of the transitive closure of Pythagoras' theorem, such that the squared distance between any two objects is less than or equal to the sum of the squared distances from the two objects to all possible third objects.

Value

An object of class 'dist'

Note

Many multivariate statistical methods are designed for euclidean spaces, and yet the direct calculation of euclidean distance is often inappropriate due to problems with joint absences. euclidify takes any dissimilarity matrix and converts it to the closest euclidean representation, generally to avoid negative eigenvalues in an eigenanalysis of the matrix.

Author(s)

David W. Roberts <droberts@montana.edu>

See Also

metrify

Examples

```
data(bryceveg) # returns a vegetation data.frame
dis.bc <- dsvdis(bryceveg,'bray/curtis') # calculate a Bray/Curtis
                                         # dissimilarity matrix
dis.euc <- euclidify(dis.bc) # calculate the nearest euclidean
                             # representation
## Not run: plot(dis.bc,dis.euc)
```

---

factorize *Change Character Vectors in Data.frames to Factors*

Description

Looks at each column in a data.frame, and converts character vector columns to factors.

Usage

```
factorize(df)
```

Arguments

df	a data.frame

Details

The function simply scans each column in a data.frame looking for character vector columns. For each character column it calls the ‘factor()’ function to convert the column to a factor.

Value

Returns a data.frame where every character column has been converted to a factor

Note

This function simplifies editing data.frames by allowing users to edit character columns (which have no levels constraints) and then converting the results to factors for modeling. It is often used in a cycle of

defactorize(df)

edit the columns as necessary to correct errors or simplify

factorize(df)

Author(s)

David W. Roberts `<droberts@montana.edu>`

See Also

defactorize

Examples

```
data(brycesite)
brycesite <- defactorize(brycesite)
brycesite$quad[brycesite$quad=='bp'] <- 'BP'
brycesite <- factorize(brycesite)
```

`gsr` *Global Search and Replace for Data.frames*

Description

Performs in-place editing of data.frames that have factor columns while correcting for the change to levels.

Usage

```
gsr(field,old,new)
```

Arguments

`field`	a vector or specific column in a data.frame
`old`	a character vector of values to search for
`new`	a character vector of values to replace the respective items in old

Details

The function temporarily converts a vector or vector column in a data.frame to a character vector, and then loops through the 'old' vector looking for values to replace with the respective value in the 'new' vector. The column is then converted back to a factor.

Value

a factor vector

Note

The function is designed to make simple editing changes to data.frames or factor vectors, resetting the levels appropriately.

Author(s)

David W. Roberts `<droberts@montana.edu>`

Examples

```
data(brycesite)
brycesite$quad <- gsr(brycesite$quad,
    old=c('bp','bc','pc','rp','tc','tr'),
    new=c('BP','BC','PC','RP','TC','TR'))
```

`hellinger` *Hellinger Data Transformation*

Description

Performs the Hellinger data transformation (square root of sample total standardized data).

Usage

```
hellinger(comm)
```

Arguments

`comm`	a community data.frame (samples as rows, species as columns)

Details

Calculates a sample total standardization (all values in a row are divided by the row sum), and then takes the square root of the values.

Value

A community data.frame

Note

Hellinger standardization is a convex standardization that simultaneously helps minimize effects of vastly different sample total abundances.

Author(s)

David W. Roberts <droberts@montana.edu>

See Also

spcmax, samptot, abundtrans

Examples

```
data(bryceveg)
hellveg <- hellinger(bryceveg)
```

---

homoteneity *Homoteneity Analysis of Classified Ecological Communities*

Description

Homoteneity is defined as 'the mean constancy of the S most constant species, expressed as a fraction, where S is the mean species richness of a type.'

Usage

```
homoteneity(comm,clustering)
```

Arguments

`comm`	a data.frame of species abundances with samples as rows and species as columns
`clustering`	a vector of (integer) class memberships, or an object of class 'clustering', class 'partana', or class `partition`

Value

A data.frame of homoteneity values

Note

This function was adapted from the Virginia Heritage Program at

http://www.dcr.virginia.gov/natural_heritage/ncstatistics.shtml

Author(s)

David W. Roberts <droberts@montana.edu>

See Also

`const`, `concov`

Examples

```
data(bryceveg) # returns a data.frame of species in sample plots
data(brycesite) # returns a data.frame of site variables
homoteneity(bryceveg,brycesite$quad) # analysis of species constancy
                                     # by USGS quad location
```

---

importance *Importance Table*

Description

For a classified set of vegetation samples, a importance table lists for each species the average or typical abundance of each species in each class.

Usage

```
importance(comm,clustering,minval=0,digits=2,show=minval,
       sort=FALSE,typical=TRUE,spcord,dots=TRUE)
```

Arguments

`comm`	a data.frame of species abundances with samples as rows and species as columns
`clustering`	a vector of (integer) class memberships, or an object of class ‘clustering’, class ‘partana’, of class `partition`
`minval`	the minimum importance a species must have in at least one class to be included in the output
`digits`	the number of digits to report in the table
`show`	the minimum value a species must have to print a value
`sort`	a switch to control interactive re-ordering
`typical`	a switch to control how mean abundance is calculated. Typical=TRUE divides the sum of species abundance by the number of plots in which it occurs; typical=FALSE divides by the number of plots in the type
`spcord`	a vector of integers to specify the order in which species should be listed in the table
`dots`	a switch to control substituting dots for small values

Value

a data.frame with species as rows, classes as columns, with average abundance of species in classes.

Note

Importance tables are often used in vegetation classification to calculate or present characteristic species for specific classes or types. Importance may be combined with `const`, `concov` and `vegtab` to achieve a vegetation table-oriented analysis.

Author(s)

David W. Roberts <droberts@montana.edu>

See Also

`const`, `vegtab`, `concov`

Examples

```
data(bryceveg) # returns a data.frame called bryceveg
data(brycesite)
class <- cut(brycesite$elev,10,labels=FALSE)
importance(bryceveg,class,minval=0.25)
```

---

indval *Dufrene-Legendre Indicator Species Analysis*

Description

Calculates the indicator value (fidelity and relative abundance) of species in clusters or types.

Usage

```
indval(x, ...)
## Default S3 method:
indval(x,clustering,numitr=1000,...)
## S3 method for class 'stride'
indval(x,comm,numitr=1,...)
## S3 method for class 'indval'
summary(object, p=0.05, type='short', digits=2, show=p,
       sort=FALSE, too.many=100, ...)
```

Arguments

`x`	a matrix or data.frame of samples with species as columns and samples as rows, or an object of class 'stride' from function `stride`
`clustering`	a vector of numeric cluster memberships for samples, or a classification object returned from `pam`, or `optpart`, `slice`, or `archi`
`numitr`	the number of randomizations to iterate to calculate probabilities
`comm`	a data.frame with samples as rows and species as columns
`object`	an object of class 'indval'
`p`	the maximum probability for a species to be listed in the summary
`type`	a switch to choose between 'short' and 'long' style summary
`digits`	the number of significant digits to show
`show`	the threshold to show values as opposed to a dot column place-holder

`sort`	a switch to control user-managed interactive table sorting
`too.many`	a threshold reduce the listing for large data sets
`...`	additional arguments to the summary or generic function

Details

Calculates the indicator value 'd' of species as the product of the relative frequency and relative average abundance in clusters. Specifically,

where:
p_{ij} = presence/absence (1/0) of species i in sample j;
x_{ij} = abundance of species i in sample j;
n_c = number of samples in cluster c;
for cluster $c \in K$;

$$f_{ic} = \frac{\sum_{j \in c} p_{ij}}{n_c}$$

$$a_{ic} = \frac{\sum_{j \in c} x_{ij}/n_c}{\sum_{k=1}^{K}(\sum_{j \in k} x_{ij}/n_k)}$$

$$d_{ic} = f_{ic} \times a_{ic}$$

Calculated on a 'stride' the function calculates the indicator values of species for each of the separate partitions in the stride.

Value

The default function returns a list of class 'indval' with components:

`relfrq`	relative frequency of species in classes
`relabu`	relative abundance of species in classes
`indval`	the indicator value for each species
`maxcls`	the class each species has maximum indicator value for
`indcls`	the indicator value for each species to its maximum class
`pval`	the probability of obtaining as high an indicator values as observed over the specified iterations

The stride-based function returns a data.frame with the number of clusters in the first column and the mean indicator value in the second.

The 'summary' function has two options. In 'short' mode it presents a table of indicator species whose probability is less then 'p', giving their indicator value and the identity of the cluster they indicate, along with the sum of probabilities for the entire data set. In 'long' mode, the indicator value of each species in each class is shown, with values less than 'show' replaced by a place-holder dot to emphasize larger values.

If 'sort==TRUE', a prompt is given to re-order the rows of the matrix interactively.

Note

Indicator value analysis was proposed by Dufrene and Legendre (1997) as a possible stopping rule for clustering, but has been used by ecologists for a variety of analyses. Dufrene and Legendre's nomenclature in the paper is somewhat ambiguous, but the equations above are taken from the worked example in the paper, not the equations on page 350 which appear to be in error. Dufrene and Legendre, however, multiply d by 100; this function does not.

Author(s)

David W. Roberts <droberts@montana.edu>

References

Dufrene, M. and Legendre, P. 1997. Species assemblages and indicator species: the need for a flexible asymmetrical approach. Ecol. Monogr. 67(3):345-366.

See Also

isamic

Examples

```
data(bryceveg) # returns a vegetation data.frame
data(brycesite)
clust <- cut(brycesite$elev,5,labels=FALSE)
summary(indval(bryceveg,clust))
```

---

isamic *Indicator Species Analysis Minimizing Intermediate Occurrences*

Description

Calculates the degree to which species are either always present or always absent within clusters or types.

Usage

```
isamic(comm,clustering,sort=FALSE)
```

Arguments

`comm`	a matrix or data.frame of samples, species as columns, samples as rows
`clustering`	a vector of numeric cluster memberships for samples, or a classification object returned from `pam`, `partana`, or `slice`
`sort`	if TRUE, return in order of highest value to lowest rather than input order

Details

Calculates the constancy (fractional occurrence of each species in every type), and then calculates twice the the sum of the absolute values of the constancy - 0.5, normalized to the number of clusters (columns).

Value

A data.frame of species indicator values

Author(s)

David W. Roberts `<droberts@montana.edu>`

References

Aho, K., D.W. Roberts, and T.W. Weaver. 2008. Using geometric and non-geometric internal evaluators to compare eight vegetation classification methods. J. Veg. Sci. 19(4):549-562.

See Also

`indval`

Examples

```
data(bryceveg)
data(brycesite)
clust <- cut(brycesite$elev,5,labels=FALSE)
isamic(bryceveg,clust)
```

`matrify` *Create Taxon Data.frames From Three Column Database Form*

Description

Takes a data.frame in three column form (sample.id, taxon, abundance) and converts it into full matrix form, and then exports it as a data.frame with the appropriate row.names and column names.

Usage

```
matrify(data, strata=FALSE, base=100)
```

Arguments

`data`	a data.frame or matrix in three column format (or database format), where the first column is the sample ID, the second column is the taxon ID, and the third sample is the abundance of that taxon in that sample.
`strata`	are the species abundances recorded in multiple strata?
`base`	what is the numeric base relative to 1.0

Details

The routine is pure R code to convert data from database form to the sparse matrix form required by multivariate analyses in packages 'labdsv' and 'vegan', as well as `dist` and other routines. If TRUE, the strata argument specifies calculating individual species abundances as independent overlap of strata. The base function is useful for converting percent to a fraction.

Value

A data.frame with samples as rows, taxa as columns, and abundance values for taxa in samples.

Note

Typically, the source of the data will be an ASCII file or a dBase database or a CSV file from an Excel file in three column format. That file can be read into a data.frame with `read.table` or `read.csv` and then that data.frame can be matrified by this function.

Author(s)

David W. Roberts <`droberts@montana.edu`>

See Also

`dematrify`

Examples

```
x <- cbind(c('a','a','b','b','b','c','c'),
           c('x','y','x','z','w','y','z'),
           c(1,2,1,3,2,2,1))
matrify(x)
```

`metrify` *Nearest Metric Space Representation of a Dissimilarity Object*

Description

Calculates the nearest metric space representation of a dissimilarity object by iterating the transitive closure of the triangle inequality rule

Usage

```
metrify(dis,upper=FALSE,diag=FALSE)
as.metric(dis,upper=FALSE,diag=FALSE)
is.metric(dis)
```

Arguments

dis	a distance or dissimilarity object returned from dist, vegdist, or dsvdis
upper	a logical switch to control whether to return the lower triangle (upper=FALSE) or upper triangle (upper=TRUE) of the distance matrix
diag	a logical switch to control whether to return the diagonal of the distance matrix

Details

Implements a constrained iteration of the transitive closure of the triangle inequality, such that the distance between any two objects is less than or equal to the sum of the distances from the two objects to a third.

Value

For metrify and as.metric, an object of class 'dist'. For is.metric returns TRUE or FALSE.

Note

Many multivariate statistical methods are designed for metric spaces, and yet the direct calculation of distance is often inappropriate due to problems with joint absences. metrify takes any dissimilarity matrix and converts it to the closest metric space representation, generally to avoid negative eigenvalues in an eigenanalysis of the matrix.

Author(s)

David W. Roberts <droberts@montana.edu>

See Also

euclidify

Examples

```
data(bryceveg) # returns a vegetation data.frame
dis.bc <- dsvdis(bryceveg,'bray/curtis') # calculate a Bray/Curtis
            #  dissimilarity matrix
dis.met <- metrify(dis.bc) # calculate the nearest euclidean
            #  representation
```

neighbors *Neighbors*

Description

Calculates the nearest neighbors in a distance/dissimilarity matrix

Usage

```
neighbors(dis,numnbr)
```

Arguments

`dis`	an object of class 'dist' such as returned by `dist`, `vegdist` or `dsvdis`
`numnbr`	the number (order) of neighbors to return

Details

For each sample unit in a dissimilarity matrix finds the 'numnbr' nearest neighbors and returns them in order.

Value

Returns a data.frame with sample units as rows and neighbors as columns, listed in order of proximity to the sample unit.

Author(s)

David W. Roberts <`droberts@montana.edu`>

Examples

```
data(bryceveg) # returns a data.frame called veg
dis.bc <- dsvdis(bryceveg,'bray/curtis')
neighbors(dis.bc,5)
```

`nmds` *Nonmetric Multidimensional Scaling*

Description

This function is simply a wrapper for the isoMDS function in the MASS package by Venables and Ripley. The purpose is to convert the output to class ‘dsvord’ to simplify plotting and additional graphical analysis as well as to provide a summary method.

Usage

```
nmds(dis,k=2,y=cmdscale(d=dis,k=k),maxit=50,trace=FALSE)
bestnmds(dis,k=2,itr=20,maxit=100,trace=FALSE,pbar=TRUE)
```

Arguments

`dis`	a dist object returned from `dist` or a full symmetric dissimilarity or distance matrix
`k`	the desired number of dimensions for the result
`y`	a matrix of initial locations (objects as rows, coordinates as columns, as many columns as specified by k). If none is supplied, `cmdscale` is used to generate them
`maxit`	the maximum number of iterations in the isoMDS routine
`trace`	a switch to control printing intermediate results
`itr`	number of random starts to find best result
`pbar`	switch to control printing progress bar in interactive sesssions

Details

The nmds function simply calls the `isoMDS` function of the MASS library, but converts the result from a list to an object of class ‘dsvord’. The only purpose for the function is to allow ‘plot’, ‘identify’, ‘surf’, and other additional methods to be defined for the class, to simplify the analysis of the result.

The ‘bestnmds’ function runs one run from a PCO solution and ‘itr-1’ number of random initial locations and returns the best result of the set.

Value

An object of class ‘dsvord’, with components:

`points`	the coordinates of samples along axes
`stress`	the "goodness-of-fit" computed as stress in percent
`type`	‘NMDS’

Note

nmds is included as part of the LabDSV package to provide a consistent interface and utility for vegetation ordination methods. Other analyses included with the same interface at present include principal components analysis (pca), principal coordinates analysis (pco), and t-distributed neighborhood embedding (t-SNE).

Author(s)

Venables and Ripley for the original isoMDS function included in the MASS package.

David W. Roberts <`droberts@montana.edu`>

References

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

Kruskal, J.B. (1964) Nonmetric multidimensional scaling: a numerical method. Psychometrics 29:115-129.

T.F. Cox and M.A.A. Cox. (1994) *Multidimensional Scaling.* Chapman and Hall.

See Also

`isoMDS` for the original function

`plot.dsvord` for the 'plot' method, the 'plotid' method to identify points with a mouse, the 'points' method to identify points meeting a logical condition, the 'hilight' method to color-code points according to a factor, the 'chullord' method to add convex hulls for a factor, or the the 'surf' method to add surface contours for continuous variables.

`initMDS` for an alternative way to automate random starts

`postMDS` for a post-solution rescaling

`metaMDS` for a full treatment of variations

Examples

```
data(bryceveg)
data(brycesite)
dis.man <- dist(bryceveg,method="manhattan")
demo.nmds <- nmds(dis.man,k=4)
plot(demo.nmds)
points(demo.nmds,brycesite$elev>8000)
plotid(demo.nmds,ids=row.names(brycesite))
```

`ordcomm` *Re-Order the Rows and Columns of a Taxon Data Frame*

Description

Allows analysts to interactively re-order a community data frame to achieve a 'structured' table following phytosociological principles.

Usage

```
ordcomm(comm,site)
```

Arguments

`comm`	a community data frame
`site`	a site or environment data frame

Details

Prints a copy of the community data frame, and then prompts for plots to move in front of another plot. It then prompts for species to move in front of a specified species. Multiple plots or species can be moved in a single move, with plot or species IDs separated by commas with no blanks. The program cycles between prompting for plots to move, and then species to move, until both prompts are responded to with blank lines.

Value

produces a list with two components:

`comm`	the new community data frame
`site`	the new site data frame

Note

This is a a fairly simple means to sort a table. For large tables, it is often possible (and preferable) to sort the tables with ordination coordinates or other indices, but this function allows analysts to order the table arbitrarily into any form.

Author(s)

David W. Roberts `<droberts@montana.edu>`

See Also

`summary.indval`,`const`,`importance`

Examples

```
## Not run: data(bryceveg) # returns a data frame called bryceveg
## Not run: data(brycesite) # returns a data frame called brycesite
## Not run: demo <- ordcomm(bryceveg,brycesite)
## Not run: newveg <- demo$taxon
## Not run: newsite <- demo$site
```

`ordcomp` *Ordination to Dissimilarity Comparison*

Description

Plots the distribution of pair-wise distances of all points in an ordination over the distances in the dissimilarity or distance matrix the ordination was calculated from. Prints the correlation between the two on the graph.

Usage

```
ordcomp(x,dis,dim,xlab="Computed Distance",
        ylab="Ordination Distance",title="",pch=1)
```

Arguments

`x`	an ordination object of class 'dsvord' from `pca`, `pco`, `nmds`, `fso` or `ordiplot`
`dis`	an object of class `dist`
`dim`	the number of dimensions in the ordination to use (default=all)
`xlab`	the X axis label for the graph
`ylab`	the Y axis label for the graph
`title`	a title for the plot
`pch`	the symbol to plot

Value

A plot is created on the current graphics device. Returns the (invisible) correlation.

Note

Ordinations are low dimensional representations of multidimensional spaces. This function attempts to portray how well the low dimensional solution approximates the full dimensional space.

Author(s)

David W. Roberts <droberts@montana.edu>

Examples

```
data(bryceveg) # produces a vegetation data.frame
dis.bc <- dsvdis(bryceveg,'bray/curtis') # creates a Bray/Curtis
                                         # dissimilarity matrix
pco.bc <- pco(dis.bc,2) # produces a two-dimensional Principal
                        # Coordinates Ordination object
ordcomp(pco.bc,dis.bc)
```

`orddist` *Ordination Point Pair-Wise Distance Calculation*

Description

Calculates the pair-wise distances of all points in an ordination. The function is simply a wrapper for the 'dist' function, but simplifies managing ordinations that store their coordinates under different names, as well as managing the desired dimensionality of the calculations.

Usage

```
orddist(x,dim)
```

Arguments

`x`	an ordination object of class 'dsvord' from `pca`, `pco`, `nmds`, `fso`
`dim`	the desired dimensionality to be included in the calculations (must be <= number of dimensions of the ordinations)

Value

An object of class 'dist' is produced

Note

Ordinations are low dimensional representations of multidimensional spaces. This function produces data on the low-dimensional distances for other analyses.

Author(s)

David W. Roberts <`droberts@montana.edu`>

Examples

```
data(bryceveg) # produces a vegetation data.frame
dis.bc <- dsvdis(bryceveg,'bray/curtis') # creates a Bray/Curtis
                                         #dissimilarity matrix
pco.bc <- pco(dis.bc,2) # produces a two-dimensional Principal
                        # Coordinates Ordination object
orddist(pco.bc,dim=2)
```

`ordneighbors` *Nearest Neighbors Plotted in Ordination Space*

Description

For each sample unit in an ordination, for each of n nearest neighbors, draws an arrow from the sample unit to its n neighbors.

Usage

```
ordneighbors(ord,dis,numnbr=1,ax=1,ay=2,digits=5,length=0.1)
```

Arguments

`ord`	an ordination object of class 'dsvord' from `pca`, `pco`, `nmds`, `fso`
`dis`	an object of class `dist`
`numnbr`	the number (order) of nearest neighbors to plot
`ax`	the dimension t plot on the X axis
`ay`	the dimension to plot on the y axis
`digits`	the number of digits to report
`length`	the length of the arrowhead

Value

Additional information is plotted on an existing ordination and summary information is printed. Returns an (invisible) list of summary values.

Note

Ordinations are low dimensional representations of multidimensional spaces. This function attempts to portray how well the low dimensional solution approximates the neighborhood relations of the full dimensional space.

If numnbr = 1 and there are ties the function plots arrows for all tied values. If n > 1 the function draws arrows for all values with rank <= n.

Author(s)

David W. Roberts `<droberts@montana.edu>`

Examples

```
data(bryceveg) # produces a vegetation data.frame
dis.bc <- dsvdis(bryceveg,'bray/curtis') # creates a Bray/Curtis
                                         # dissimilarity matrix
pco.bc <- pco(dis.bc,2) # produces a two-dimensional Principal
                        # Coordinates Ordination object
plot(pco.bc)
ordneighbors(pco.bc,dis.bc)
```

`ordpart` *Ordination Partitioning*

Description

This function allows users to partition or classify the points in an ordination by identifying clusters of points with a mouse

Usage

```
ordpart(ord, ax = 1, ay = 2)
```

Arguments

`ord`	an ordination of class 'dsvord' produced by nmds, pco, pca or other labdsv ordination functions
`ax`	the first axis number in the ordination plot
`ay`	the second axis number in the ordination plot

Details

Given a plot of an ordination, you assign plots to clusters by drawing a polygon with the first mouse button to include all points in a given cluster. To end that cluster, click the right mouse button to close the polygon. Plots included in that cluster will be color-coded to indicate membership. Start the next cluster by drawing another polygon. To end, click the right mouse button again after closing the last polygon. Plots within more than one polygon are assigned membership in the last polygon which includes them; plots which are not within any polygon are assigned membership in cluster zero.

Value

A integer vector of cluster membership values

Note

Although the routine could easily be adapted for any scatter plot, it is currently only designed for objects of class 'dsvord'.

Author(s)

David W. Roberts <droberts@montana.edu>

Examples

```
data(bryceveg)
data(brycesite)
dis.bc <- dsvdis(bryceveg,'bray/curtis')
nmds.1 <- nmds(dis.bc,5)
plot(nmds.1)
## Not run: clustering <- ordpart(nmds.1)
```

`ordtest` *Ordination Distribution Test*

Description

Testing the distribution of points in an ordination

Usage

```
ordtest(ord, var, dim=1:ncol(ord$points), index = 'euclidean',
   nitr = 1000)
```

Arguments

`ord`	an object of class ‘dsvord’
`var`	a logical or factor vector used to organize the calculation of within-set distances
`dim`	the number of dimensions to use in the calculation
`index`	the distance metric for the calculation of within-set distances. Currently only euclidean is accepted
`nitr`	the number of iterations to perform to establish p-values

Details

Calculates the sum of within-set pair-wise distances and compares to ‘nitr’ permutations of the same distribution to calculate the probability of observing clusters as tight as observed or tighter. The p-value is calculated by running nitr-1 permutations and counting the number of cases where the sum of pair-wise distances is as small as smaller than observed. That count is increased by one and divided by nitr to estimate p.

Value

Produces a list with components:

`obs`	the observed sum of within-set distances
`p`	the probability of obtaining a value that small
`reps`	the sum of within-set pairwise distances for all permutations

Author(s)

David W. Roberts <droberts@montana.edu>

See Also

anosim

Examples

```
data(bryceveg)
data(brycesite)
dis.bc <- dsvdis(bryceveg,'bray/curtis')
pco.bc <- pco(dis.bc)
plot(pco.bc)
demo <- ordtest(pco.bc,brycesite$quad)
demo$p
```

---

pca *Principal Components Analysis*

Description

Principal components analysis is a eigenanalysis of a correlation or covariance matrix used to project a high-dimensional system to fewer dimensions.

Usage

```
pca(mat, cor = FALSE, dim = min(nrow(mat),ncol(mat)))
## S3 method for class 'pca'
summary(object, dim = length(object$sdev), ...)
## S3 method for class 'pca'
scores(x, labels = NULL, dim = length(x$sdev), ...)
## S3 method for class 'pca'
loadings(x, dim = length(x$sdev), digits = 3, cutoff = 0.1, ...)
## S3 method for class 'pca'
varplot(x, dim=length(x$sdev),...)
```

Arguments

`mat`	a matrix or data.frame of interest, samples as rows, attributes as columns
`cor`	logical: whether to use a correlation matrix (if TRUE), or covariance matrix (if FALSE)
`dim`	the number of dimensions to return
`object`	an object of class ‘pca’
`x`	an object of class ‘dsvord’ and type=’pca’
`labels`	an (optional) vector of labels to identify points

`digits`	number of digits to report
`cutoff`	threshold to suppress printing small values
`...`	arguments to pass to function summary or graphics arguments

Details

PCA is a common multivariate technique. The version here is simply a wrapper for the `prcomp` function to make its use and plotting consistent with the other LabDSV functions.

Value

an object of class "pca", a list with components:

`scores`	a matrix of the coordinates of the samples in the reduced space
`loadings`	a matrix of the contributions of the variables to the axes of the reduced space.
`sdev`	a vector of standard deviations for each dimension

Note

The current version of pca is based on the `prcomp` function, as opposed to the `princomp` function. Nonetheless, it maintains the more conventional labels "scores" and "loadings", rather than `x` and rotation. prcomp is based on a singular value decomposition algorithm, as has worked better in my experience. In the rare cases where it fails, you may want to try `princomp`.

Author(s)

David W. Roberts <`droberts@montana.edu`>

See Also

`princomp`, `prcomp`, `pco`, `nmds`, `fso`, `cca`

Examples

```
data(bryceveg) # returns a vegetation data.frame
data(brycesite)
x <- pca(bryceveg,dim=10)  # returns the first 10 eigenvectors
                           # and loadings
plot(x)
surf(x,brycesite$elev)
points(x,brycesite$depth=='deep')
```

pco *Principal Coordinates Analysis*

Description

Principal coordinates analysis is an eigenanalysis of distance or metric dissimilarity matrices.

Usage

```
pco(dis, k=2)
```

Arguments

`dis`	the distance or dissimilarity matrix object of class "dist" returned from `dist`, `vegdist`, or `dsvdis`
`k`	the number of dimensions to return

Details

pco is simply a wrapper for the `cmdscale` function of Venebles and Ripley to make plotting of the function similar to other LabDSV functions

Value

An object of class ‘pco’ with components:

`points`	the coordinates of samples on eigenvectors

Note

Principal Coordinates Analysis was pioneered by Gower (1966) as an alternative to PCA better suited to ecological datasets.

Author(s)

of the ‘cmdscale’ function: Venebles and Ripley

of the wrapper function David W. Roberts <`droberts@montana.edu`>

References

Gower, J.C. (1966) Some distance properties of latent root and vector methods used in multivariate analysis. Biometrika 53:325-328.

See Also

`cmdscale`, `pca`, `nmds`, `cca`

Examples

```
data(bryceveg) # returns a vegetation data.frame
dis.bc <- dsvdis(bryceveg,'bray/curtis')
                  # returns an object of class dist'
veg.pco <- pco(dis.bc,k=4) # returns first 4 dimensions
plot(veg.pco)
```

`plot.dsvord` *Plotting Routines For LabDSV Ordinations*

Description

A set of routines for plotting, highlighting points, or adding fitted surfaces to ordinations.

Usage

```
## S3 method for class 'dsvord'
plot(x, ax = 1, ay = 2, col = 1, title = "", pch = 1,
                        xlab = paste(x$type, ax), ylab = paste(x$type, ay), ...)
## S3 method for class 'dsvord'
points(x, which, ax = 1, ay = 2, col = 2, pch = 1, cex = 1,
                         breaks=FALSE, ...)
## S3 method for class 'dsvord'
plotid(ord, ids = seq(1:nrow(ord$points)), ax = 1, ay = 2,
        col = 1, ...)
## S3 method for class 'dsvord'
hilight(ord, overlay, ax = 1, ay = 2, title="",
         cols=c(2,3,4,5,6,7), glyph=c(1,3,5), ...)
## S3 method for class 'dsvord'
chullord(ord, overlay, ax = 1, ay = 2, cols=c(2,3,4,5,6,7),
         ltys = c(1,2,3), ...)
## S3 method for class 'dsvord'
ellip(ord, overlay, ax = 1, ay = 2, cols=c(2,3,4,5,6,7),
         ltys = c(1,2,3), ...)
## S3 method for class 'dsvord'
surf(ord, var, ax = 1, ay = 2, thinplate = TRUE, col = 2,
         labcex = 0.8, lty = 1, family = gaussian, gamma=1, grid=50, ...)
## S3 method for class 'dsvord'
thull(ord,var,grain,ax=1,ay=2,col=2,grid=51,nlevels=5,
         levels=NULL,lty=1,
      numitr=100,...)
## S3 method for class 'dsvord'
density(ord, overlay, ax = 1, ay = 2, cols = c(2, 3, 4, 5,
    6, 7), ltys = c(1, 2, 3), numitr, ...)
```

Arguments

`x`	an object of class ‘dsvord’
`ax`	the dimension to use for the X axis
`ay`	the dimension to use for the Y axis
`title`	a title for the plot
`xlab`	label for X axis
`ylab`	label for Y axis
`which`	a logical variable to specify points to be highlighted
`breaks`	a logical switch to control using variable glyph sizes in ‘points’
`ord`	an object of class ‘dsvord’
`overlay`	a factor or integer vector to hilight or distinguish
`cols`	the sequence of color indices to be used
`glyph`	the sequence of glyphs (pch) to be used
`lty`	the line type to be used
`ltys`	the sequence of line types to be used
`var`	a variable to be surfaced or tension hulled
`thinplate`	a logical variable to control the fitting routine: thinplate=TRUE (the default) fits a thinplate spline, thinplate=FALSE fits independent smooth splines. If you have too few data points you may have to specify thinplate=FALSE
`family`	controls the link function passed to ‘gam’: one of ‘gaussian’, ‘binomial’, ‘poisson’ or ‘nb’
`gamma`	controls the smoothness of the fit from `gam`
`grid`	the number of X and Y values to use in establishing a grid for use in surf
`grain`	the size of cell to use in calculating the tensioned hull
`nlevels`	the number of contours to draw in representing the tensioned hull
`ids`	identifier labels for samples. Defaults to 1:n
`col`	color index for points or contours
`labcex`	size of contour interval labels
`pch`	plot character: glyph to plot
`cex`	character expansion factor: size of plotted characters
`numitr`	the number of iterations to use in estimating the probability of the observed density
`levels`	specific levels for contours in thull
`...`	arguments to pass to the plot function

Details

Function 'plot' produces a scatter plot of sample scores for the specified axes, erasing or over-plotting on the current graphic device. Axes dimensions are controlled to produce a graph with the correct aspect ratio. Functions 'points', 'plotid', and 'surf' add detail to an existing plot. The axes specified must match the underlying plot exactly.

Function 'plotid' identifies and labels samples (optionally with values from a third vector) in the ordination, and requires interaction with the mouse: left button identifies, right button exits.

Function 'points' is passed a logical vector to identify a set of samples by color of glyph. It can be used to identify a single set meeting almost any criterion that can be stated as a logical expression.

Function 'hilight' is passed a factor vector or integer vector, and identifies factor values by color and glyph.

Function 'chullord' is passed a factor vector or integer vector, and plots a convex hull around all points in each factor class. By specifying values for arguments 'cols' and 'ltys' it is possible to control the sequence of colors and linetypes of the convex hulls.

Function 'ellip' is passed a factor vector or integer vector, and plots minimal volume ellipses containingg all points within a class. By specifying values for arguments 'cols' and 'ltys' it is possible to control the sequence of colors and linetypes of the ellipses.

Function 'density' calculates the fraction of points within the convex hull that belong to the specified type.

Function 'surf' calculates and plots fitted surfaces for logical or quantitative variables. The function employs the `gam` function to fit a variable to the ordination coordinates, and to predict the values at all grid points. The grid is established with the 'expand.grid' function, and the grid is then specified in a call to 'predict.gam'. The predicted values are trimmed to the the convex hull of the data, and the contours are fit by 'contour'. The default link function for fitting the GAMs is 'gaussian', suitable for unbounded continuous variables. For logical variables you should specify 'family = binomial' to get a logistic GAM, and for integer counts you should specify 'family = poisson' to get a Poisson GAM or 'family='nb'' to get a negative binomial fit.

Function 'thull' calculates a tensioned hull for a specific variable on the ordination. A tensioned hull is a minimum volume container. The grain size must be specified as a fraction of the units of the NMDS, with larger values generating smoother representations, and smaller numbers a more resolved container. 'thull' returns an invisible object of class 'thull' which has an associated plot function. Plotting the thull object produces a colored surface representation of the thull with optional contour lines.

Value

Function 'plotid' returns a vector of row numbers of identified plots

Note

The contouring routine using `predict.gam` follows `ordisurf` as suggested by Jari Oksanen.

Author(s)

David W. Roberts <`droberts@montana.edu`>

Examples

```
data(bryceveg)
data(brycesite)
dis.bc <- dsvdis(bryceveg,'bray/curtis')
nmds.1 <- nmds(dis.bc,5)
plot(nmds.1)
points(nmds.1,brycesite$elev>8000)
surf(nmds.1,brycesite$elev)
## Not run: plotid(nmds.1,ids=row.names(bryceveg))
```

`plot.thull` *Plotting a Tensioned Hull*

Description

A tensioned hull is a minimum volume container for specified elements of an ordination. A 'thull' object is returned as an invisible object by plotting a thull of an NMDS or PCO (or MFSO). Subsequently plotting the returned thull results in an 'image' of the representation.

Usage

```
## S3 method for class 'thull'
plot(x,col=rainbow(20),levels=NULL,cont=TRUE,
          xlab=x$xlab,ylab=x$ylab,main=x$main,...)
```

Arguments

`x`	an object of class 'thull' from function thull
`col`	the color to use plotting the contours
`levels`	the specific levels desired for the contours
`cont`	a logical variable to control plotting contours on the image representation of the tensioned hull
`xlab`	the X axis label
`ylab`	the Y axis label
`main`	the main title
`...`	other graphics parameters

Details

Tensioned hull analysis fits a minimum volume envelope to specific points in an ordination. A tensioned hull object is returned from function `thull` of a ordination of class 'dsvord'. This function plots the resulting tensioned hull as an image, with optional overlays of contours.

Value

Produces a plot on the current graphic device.

Author(s)

David W. Roberts <droberts@montana.edu>

Examples

```
data(bryceveg) # returns a data.frame called bryceveg
dis.bc <- dsvdis(bryceveg,'bray') # calculates a Bray-Curtis
                                  # dissimilarity matrix
nmds.bc <- nmds(dis.bc) # calculates an NMDS ordination
plot(nmds.bc) # plots the ordination on the current device
demo.thull <- thull(nmds.bc,bryceveg$arcpat,0.25) # calculates
                        # the tensioned hull representing the
                        # distributtion of a species
plot(demo.thull) # portrays the image version of the tensioned hull
```

---

predict *Predict species abundances in an ordination*

Description

This function fits a Generalized Additive Model (GAM) for each species in a data.frame against an ordination.

Usage

```
## S3 method for class 'dsvord'
predict(object,comm,minocc=5,dims=1:ncol(object$points),
                            family='nb',gamma=1,keep.models=FALSE,...)
```

Arguments

`object`	an object of class dsvord
`comm`	a community matrix or data.frame with samples as rows and species as columns
`minocc`	the minimum number of occurrences to model a species
`dims`	which specific dimensions to include
`family`	the error distribution specifier for the GAM function; can be 'nb' for negative binomial, 'poisson' for the Poisson distribution, or 'binomial' for presence/absence data
`gamma`	the gamma parameter to control fitting GAM models
`keep.models`	a switch to control saving the individual GAM models
`...`	ancillary arguments to function predict

Details

The predict function sequentially and independently fits a GAM model of each species distribution as a function of ordination coordinates, using the family and gamma specifiers supplied in the function call, or their defaults. The function fits two or three dimensional models; if the length of dims is greater than three the dimensions are truncated to the first three chosen.

Value

A list object with vector elements aic, dev.expl, adj.rsq, and matrix fitted. Optionally, if keep.models is TRUE, a list with all of the GAM models fitted. list element aic gives the model AICs for each species, dev.expl gives the deviance explained, adj.rsq gives the adjusted r-Squared, and fitted gives the expected abundance of each species in each sample unit.

Author(s)

David W. Roberts `<droberts@montana.edu>`

See Also

calibrate for the complementary function that fits GAM models for environment variables

Examples

```
data(bryceveg)
dis.man <- dist(bryceveg,method="manhattan")
demo.nmds <- nmds(dis.man,k=4)
## Not run: res <- predict(demo.nmds,bryceveg,minocc=10)
```

`raretaxa` *Identify Rare Taxa in a Data Set*

Description

Identifies the distribution of rare taxa in a community data.frame, using a specified rareness threshold.

Usage

```
raretaxa(comm,min=1,log=FALSE,type='b', panel='all')
```

Arguments

`comm`	a community data.frame with samples as rows and species as columns
`min`	the minimum number of occurrences for a species to be considered rare
`log`	controls whether or not the Y axis on some graphs should be log scaled
`type`	the plot type. ‘b’ = both points and lines
`panel`	a switch to control which graphic is displayed. Can be either an integer from 1 to 3 or the word ‘all’.

Details

Rare species are an issue in ecological data sets. This function produces three graphs identifying (1) the distribution of rare species/plot, (2) the mean abundance (when present) of rare species, and (3) the total abundance or rare species/plot.

Value

Produces only graphs and returns no output

Author(s)

David W. Roberts <droberts@montana.edu>

Examples

```
data(bryceveg)
raretaxa(bryceveg,min=3,log=TRUE)
```

---

reconcile *Reconcile Community and Site Data.Frames*

Description

reconcile takes two data frames (comm and site) and sorts both into the same order, and then deletes any rows unique to either of the two data.frames, achieving perfect correspondence of the two.

Usage

```
reconcile(comm,site,exlist)
```

Arguments

`comm`	a community abundance data.frame with samples as rows and species as columns
`site`	a data.frame of site or environmental variables with samples as rows and variables as columns
`exlist`	a switch to control listing specific plots vs simply the number of plots

Details

reconcile sorts each data.frame alphabetically by row.name, and then compares the list of row.names to identify sample plots common to both data.frames. Sample plots which occur in only one of the data.frames are deleted.

Value

A list object with two elements: comm and site, which are the sorted and reconciled data.frames.

Note

Package labdsv (and many other packages in ecological data analysis) require two data.frames to structure the data. One contains the abundance of species within samples with samples as rows and species as columns. This data.frame I refer to as the sQuotecomm data.frame. The other data.frame contains all the environmental or site data collected at the same samples. This data.frame I refer to as the 'site' data.frame. Due to independent subsampling, sorting or editing of the data (often outside of R) the two data.frames often lose the necessary requirement of the identical number of rows, with the rows in exactly the same order. The reconcile() function is a simple remedy to correct this situation while maintaining the maximum amount of data.

Author(s)

David W. Roberts <droberts@montana.edu>

Examples

```
data(bryceveg)   # returns a data.frame of taxon abundance
data(brycesite)  # returns a data.frame of site variables
test <- reconcile(bryceveg,brycesite)
```

`rndcomm` *Randomize a Community Data.Frame*

Description

Permutes a vegetation (or other) data.frame to establish a basis for null model tests in vegetation ecology.

Usage

```
rndcomm(comm,replace=FALSE,species=FALSE,plots=FALSE)
```

Arguments

`comm`	the vegetation (or other taxon) data.frame, samples as rows, species as columns
`replace`	a switch for permuting (if FALSE) or boostrapping (if TRUE)
`species`	a switch to control randomizing by species (if TRUE), maintaining species occurrence distributions
`plots`	a switch to control randomizing by samples (if TRUE), maintaining plot-level species richness

Details

Permutes or bootstraps a vegetation data frame for input to `dist`, `vegdist`, `dsvdis`, or other routines. Can randomize by columns (species=TRUE), samples (plots=TRUE), or fully (neither species nor plots = TRUE).

Value

a data.frame with samples as rows and species as columns of the same dimensions as entered.

Note

Randomizing vegetation often leads to unrealistic data distributions, but this function attempts to preserve either species occurrence distributions or plot-level species richness. It is probably worth examining the output of this function with `abuocc` to see its characteristics before engaging in extensive analysis.

Author(s)

David W. Roberts <`droberts@montana.edu`>

Examples

```
data(bryceveg) # returns a vegetation data.frame called bryceveg
test <- rndcomm(bryceveg,species=TRUE) # preserves species abundance
                                        # distribution
test2 <- rndcomm(bryceveg,plots=TRUE) # preserves plot-level
                                       # species richness
```

`rnddist` *Random Distance*

Description

Calculates a random distance matrix for use in null model analysis.

Usage

```
rnddist(size, method='metric', sat = 1.0, upper=FALSE,
        diag=FALSE)
```

Arguments

`size`	the number of items to calculate the distances for
`method`	the desired properties of the matrix. Must be either ‘metric’ or ‘euclidean’
`sat`	a saturation coefficient to set an upper limit less than 1.0 that truncates maximum values to simulate a dissimilarity rather than a distance
`upper`	logical: whether to print the upper triangle (default=FALSE)
`diag`	logical: whether to print the diagonal (default=FALSE)

Details

Generates a matrix of $size^2$ uniform random numbers and passes the matrix to `metrify` or `euclidify` to ensure the metric or euclidean properties of the distances. Values are normalized to a maximum of 1.0.

Value

A dissimilarity object of class 'dist'

Author(s)

David W. Roberts <droberts@montana.edu>

See Also

metrify, euclidify

Examples

```
x <- rnddist(100)
pco.x <- pco(x)
plot(pco.x)
```

---

samptot *Sample total standardization*

Description

Standardizes a community data set to a sample total standardization.

Usage

```
samptot(comm)
```

Arguments

comm	a community matrix (samples as rows, species as columns)

Details

This function simply calculates row sums for the community matrix and then divides all values in that row by the appropriate sum so that all samples total to 1.0.

Value

A data frame of sample total standardized community data.

Author(s)

David W. Roberts <droberts@montana.edu>

See Also

spcmax, abundtrans

Examples

```
data(bryceveg)
stveg <- samptot(bryceveg)
apply(stveg,1,sum)
```

`spcdisc` *Species Discrimination Analysis*

Description

Calculates the degree to which species are restricted to certain classes of classified vegetation

Usage

```
spcdisc(x,sort=FALSE)
```

Arguments

`x`	a classified vegetation table returned by 'const', or 'importance'
`sort`	return in sorted order if TRUE

Details

Calculates a Shannon-Weiner information statistic on the relative abundance of species within classes.

Value

A vector of discrimination values.

Author(s)

David W. Roberts <`droberts@montana.edu`>

See Also

`const`, `importance`, `indval`, `isamic`

Examples

```
data(bryceveg)
data(brycesite)
test <- const(bryceveg,brycesite$quad)
spcdisc(test)
```

`spcmax` *Species Maximum Standardization*

Description

Standardizes a community data.frame by dividing the abundance of each species by the maximum value obtained for that species.

Usage

```
spcmax(comm)
```

Arguments

`comm`	community data.frame (samples as rows, species as columns)

Details

This is a simple standardization to make each species abundance scaled from 0 to 1, essentially relativizing abundance by species and making each species equal in the calculation of distance or dissimilarity or other analyses.

Value

A data.frame of standardized community data.

Author(s)

David W. Roberts `<droberts@montana.edu>`

See Also

samptot, abundtrans, hellinger

Examples

```
data(bryceveg)
smveg <- spcmax(bryceveg)
apply(smveg,2,max)
```

`stepdist` *Step-Across Distance*

Description

Solves for the shortest-path step-across distance for a given distance matrix

Usage

```
stepdist(dis,alpha)
```

Arguments

`dis`	a distance or dissimilarity object of class 'dist'
`alpha`	a threshold distance to establish the step-across

Details

The function takes the dist object and converts all values >= alpha to 9999.9 and then solves for new distances by calculating the transitive closure of the triangle inequality.

Value

an object of class 'dist'

Note

The 'dsvdis' function includes a step-across function in the initial calculation of a distance or dissimilarity matrix. This function simply allows the conversion to take place at a later time, or on distance metrics that 'dsvdis' doesn't support.

Author(s)

David W. Roberts <`droberts@montana.edu`>

Examples

```
data(bryceveg)
dis.bc <- dsvdis(bryceveg,'bray')
dis.bcx <- stepdist(dis.bc,1.00)
disana(dis.bcx)
```

`tsne`	*t-Distributed Stochastic Neighbor Embedding*

Description

This function is a wrapper for the Rtsne function in the Rtsne package by Krijthe and van der Maaten. The purpose is to convert the output to class 'dsvord' to simplify plotting and additional graphical analysis as well as to provide a summary method.

Usage

```
tsne(dis,k=2,perplexity=30,theta= 0.0,eta=200)
besttsne(dis,k=2,itr=100,perplexity=30,theta=0.0,eta = 200,pbar=TRUE)
```

Arguments

`dis`	a dist object returned from `dist` or a full symmetric dissimilarity or distance matrix
`k`	the desired number of dimensions for the result
`perplexity`	neighborhood size parameter (should be less than (size(dis)-1) /3
`theta`	Speed/accuracy trade-off; set to 0.0 for exact TSNE, (0,0,0.5] for increasing speeed (default: 0.0)
`eta`	Learning rate
`itr`	number of random starts to find best result
`pbar`	switch to control printing progress bar in interactive sessions

Details

The tsne function simply calls the `Rtsne` function of the Rtsne package with a specified distance/dissimilarity matrix rather than the community matrix. By convention, t-SNE employs a PCA on the input data matrix, and calculates distances among the first 50 eigenvectors of the PCA. Rtsne, however, allows the submission of a pre-calculated distance/dissimilarity matrix in place of the PCA. Given the long history of research into the use of PCA in ecological community analysis, tsne allows the simple use of any of a vast number of distance/dissimilarity matrices known to work better with ecological data.

In addition, the tsne function converts the output to an object of class 'dsvord' to simplify plotting and analyses using the many functions defined for objects of class 'dsvord'. (see `plot.dsvord` for more details.)

The 'besttsne' function runs one run from a PCO solution as the initial configuration and 'itr-1' number of random initial locations and returns the best result of the set.

Value

an object of class 'dsvord', with components:

`points`	the coordinates of samples along axes
`type`	't-SNE'

Note

tsne is included as part of the LabDSV package to provide a consistent interface and utility for ecological community ordination methods. Other analyses included with the same interface at present include nonmetric multidimensional scaling (NMDS), principal components analysis (pca), and principal coordinates analysis (pco).

Author(s)

Jesse H. Krijthe for the original Rtsne R code, adapted from C++ code from Laurens van der Maaten.

David W. Roberts <droberts@montana.edu>

References

van der Maaten, L. 2014. Accelerating t-SNE using Tree-Based Algorithms. Journal of Machine Learning Research, 15, p.3221-3245.

van der Maaten, L.J.P. & Hinton, G.E., 2008. Visualizing High-Dimensional Data Using t-SNE. Journal of Machine Learning Research, 9, pp.2579-2605.

Krijthe, J,H, 2015. Rtsne: T-Distributed Stochastic Neighbor Embedding using a Barnes-Hut Implementation, URL: https://github.com/jkrijthe/Rtsne

See Also

`Rtsne` for the original function

`plot.dsvord` for the 'plot' method, the 'plotid' method to identify points with a mouse, the 'points' method to identify points meeting a logical condition, the 'hilight' method to color-code points according to a factor, the 'chullord' method to add convex hulls for a factor, or the the 'surf' method to add surface contours for continuous variables.

Examples

```
data(bryceveg)
data(brycesite)
dis.man <- dist(bryceveg,method="manhattan")
demo.tsne <- tsne(dis.man,k=2)
plot(demo.tsne)
points(demo.tsne,brycesite$elev>8000)
plotid(demo.tsne,ids=row.names(brycesite))
```

`vegtab` *Vegetation Table*

Description

Produces an ordered table of abundance of species in samples, sub-sampled by (an optional) classification of the samples

Usage

```
vegtab(comm,set,minval=1,pltord,spcord,pltlbl,trans=FALSE)
```

Arguments

`comm`	a vegetation (or other taxon) data.frame
`set`	a logical variable specifying which samples to include
`minval`	a minimum abundance threshold to include in the table
`pltord`	a numeric vector specifying the order of rows in the output
`spcord`	a numeric vector specifying the order of columns in the output
`pltlbl`	a vector specifying an alternative row label (must be unique!)
`trans`	a logical variable to control transposing the table

Details

Subsets a vegetation data.frame according to specified plots or minimum species abundances, optionally ordering in arbitrary order.

Value

a data.frame with specified rows, columns, and row.names

Note

Vegetation tables are a common tool in vegetation analysis. In recent years analysis has tended to become more quantitative, and less oriented to sorted tables, but even still presenting the results from these analyses often involves a sorted vegetation table.

Author(s)

David W. Roberts `<droberts@montana.edu>`

See Also

`vegemite`

Examples

```
data(bryceveg)  # returns a vegetation data frame called bryceveg
data(brycesite) # returns an environmental data frame called
                # brycesite
vegtab(bryceveg,minval=10,pltord=brycesite$elev)
        # produces a sorted table for species whose abundance sums
        # to 10, with rows in order of elevation.
```

Index