Package ‘fossil’

March 23, 2020

Type Package

Title Palaeoecological and Palaeogeographical Analysis Tools

Version 0.4.0

Date 2020-03-20

Author Matthew J. Vavrek <matthew@matthewvavrek.com>

Maintainer Matthew J. Vavrek <matthew@matthewvavrek.com>

Depends sp, maps, shapefiles

Description A set of analytical tools useful in analysing ecological and geographical data sets, both ancient and modern. The package includes functions for estimating species richness (Chao 1 and 2, ACE, ICE, Jacknife), shared species/beta diversity, species area curves and geographic distances and areas.

License GPL (>= 2)

URL http://matthewvavrek.com/programs-and-code/fossil/

NeedsCompilation no

Repository CRAN

Date/Publication 2020-03-23 11:30:05 UTC

R topics documented:

`fossil-package` *fossil: Palaeoecological and Palaeogeographical Analysis Tools*

Description

A set of analytical tools useful in analysing ecological and geographical data sets, both ancient and modern. The package includes functions for estimating species richness (Chao 1 and 2, ACE, ICE, Jacknife), shared species/beta diversity, species area curves and geographic distances.

Details

The fossil package is designed to be used by palaeoecologists and palaeobiogeographers, providing to them a set of useful tools including species similarity indices, species estimators, minimum spanning tree/forest functions, and an assortment of other useful tools.

Author(s)

Matthew Vavrek <matthew@matthewvavrek.com>

ACE — *Abundance- and Incidence-based Coverage Estimators*

Description

Computes the extrapolated species richness of a population using the Abundance- and Incidence-based Coerage Estimators

Usage

```
ACE(x, taxa.row = TRUE)
ICE(x, taxa.row = TRUE)
```

Arguments

`x`	a vector, matrix or data frame of positive integers or zero of any size
`taxa.row`	whether each row of the matrix is a different taxon; if so, value is T

Details

These functions compute the ACE and ICE richness estimators, respectively. Both functions will accept a vector, matrix or data frame of any size made up of positive integers and zeros. Matrices are by default treated such that each row is a different taxon and each column is a sample or locality, however if they are arranged with the taxa as columns, change the argument `taxa.row` to FALSE Take note that `ACE` is intended only for use with abundance data, and not presence absence data. While `ICE` will accept abundance matrices, it will internally convert the matrix to presence absence data. Note that if `ACE` returns NaN or Inf as a value, that Chao1 will be used in it's place as per the recommendation made by Colwell in EstimateS.

Value

A value representing a minimum number of species present in the assemblage if the entire population were to be censused.

Author(s)

Matthew Vavrek, with recommendations from the EstimateS reference manual by R.K. Colwell

References

Chao, A., M.-C. Ma, & M. C. K. Yang. 1993. Stopping rules and estimation for recapture debugging with unequal failure rates. Biometrika 80, 193-201.

Chao, A., W.-H. Hwang, Y.-C. Chen, and C.-Y. Kuo. 2000. Estimating the number of shared species in two communities. Statistica Sinica 10:227-246.

Chazdon, R. L., R. K. Colwell, J. S. Denslow, & M. R. Guariguata. 1998. Statistical methods for estimating species richness of woody regeneration in primary and secondary rain forests of NE Costa Rica. Pp. 285-309 in F. Dallmeier and J. A. Comiskey, eds. Forest biodiversity research, monitoring and modeling: Conceptual background and Old World case studies. Parthenon Publishing, Paris.

See Also

For related species estimators, see chao1, bootstrap and jack1, and spp.est to calculate multiple indices at once.

Examples

```
## sample vector
a<-c(0,5,1,1,2,0,0,1,0,0,8,45)
ACE(a)


## matrix format
a<-matrix(c(0,5,1,1,2,0,0,1,0,0,8,45),4,3)
ACE(a)
ICE(a)

## presence absence matrix
a<-matrix(c(0,1,1,1,1,0,0,1,0,0,1,1),4,3)
ACE(a)
ICE(a)
```

---

aic.nest *Nestedness of samples using AIC*

Description

Test if two empirical samples are drwan from the same or different communities

Usage

```
    aic.nest(comm1, comm2, base=exp(1))
```

Arguments

comm1, comm2	lists of abundance data from two empirical samples
base	base of the log used in the calculation of Shannon's diversity

Details

This function tests if two empirical samples are drawn from the same community, based on the AIC scores.

Value

returns two AIC scores, the first assuming the two empirical samples are drawn from the same distribution, the other assuming the two empirical samples are from different distributions

Author(s)

Matthew Vavrek

See Also

`simpson`

Examples

```
#for example, two different communities
a<-c(12,4,12,1,4,0,6,5,0,0,0)
b<-c(0,11,4,3,6,7,7,2,23,5,8)

#if the aic score is lower, it is the better model
aic.nest(a,b)

#from the same community
a<-c(5,6,5,6,5,6,5,6,5,2,1,1)
b<-c(2,3,2,3,2,3,2,3,2,1,0,0)
aic.nest(a,b)
```

`bootstrap` *Bootstrap Species Richness Estimator*

Description

Computes the bootstrap species richness estimator for abundance or presence-absence data

Usage

```
bootstrap(x, taxa.row = TRUE, abund = TRUE, samples = NA)
```

Arguments

`x`	a vector, matrix or data frame of positive integers or zero of any size
`taxa.row`	whether each row of the matrix is a different taxon
`abund`	whether the input is abundance (or presence/absence) based
`samples`	if input is a vector file, the number of samples must be included

Details

The bootstrap estimator

Value

Returns a single value for the Bootstrap Species Estimator

Author(s)

Matthew Vavrek

References

Smith, E.P. & van Belle, G. 1984. Nonparametric estimation of species richness. Biometrics 40, 119-129.

See Also

`jack1`, `ACE`, `chao1`

Examples

```
## sample vector
a<-c(0,5,1,1,2,0,0,1,0,0,8,45)
bootstrap(a,samples=45)

## matrix format
a<-matrix(c(0,5,1,1,2,0,0,1,0,0,8,45),4,3)
bootstrap(a)
bootstrap(a,,FALSE)


## presence absence matrix
a<-matrix(c(0,1,1,1,1,0,0,1,0,0,1,1),4,3)
bootstrap(a,,FALSE)
```

`chao.sd` *Chao's estimation of standard error*

Description

Computes the standard error for `chao1` or `chao2`

Usage

```
chao.sd(x)
```

Arguments

`x`	a vector of abundances or frequencies of occurrences

Details

primarily designed to be used internally by `spp.est` to calculate the errors for the chao estimators

Value

returns a value for standard deviation for `chao1` or `chao2`

Author(s)

Matthew Vavrek

References

Colwell, R.K. 2010. EstimateS: Statistical estimation of species richness and shared species from samples. Version 8.2. User's Guide and application published at: http://purl.oclc.org/estimates.

See Also

`chao1`, `spp.est`

Examples

```
## sample vector
a<-c(0,5,1,1,2,0,0,1,0,0,8,45)
chao.sd(a)
```

`chao.sorenson` *Chao's Jaccard and Sorenson Estimators of Shared Species*

Description

Chao's Jaccard and Sorenson shared species estimators for use with incomplete datasets

Usage

```
chao.sorenson(x, y)
chao.jaccard(x, y)
```

Arguments

`x`	species from group A
`y`	species from group B

Details

You must provide two separate vectors, with species arranged in the same order, from area A and B. If species are present in one site but not the other, these must be recorded for both sites; the site where they are not found should be coded as a zero. Species not present at either site are ignored.

Value

Returns the Chao-Jaccard or Chao-Sorenson similarity index for the two sites in question.

Author(s)

Matthew Vavrek

References

Chao, A., R. L. Chazdon, et al. 2005. A new statistical approach for assessing similarity of species composition with incidence and abundance data. Ecology Letters 8: 148-159.

See Also

`bray.curtis`

Examples

```
##Species counts from two different locations
a <- c(1,0,4,3,5,0,0,7)
b <- c(2,1,3,0,0,1,0,6)
chao.sorenson(a,b)
chao.jaccard(a,b)
```

`chao1` *Chao's Species Estimators*

Description

Computes the Chao species estimator for abundance or presence-absence data

Usage

```
chao1(x, taxa.row = TRUE)
chao2(x, taxa.row = TRUE)
```

Arguments

`x`	a vector, matrix or data frame with species by samples
`taxa.row`	a logical argument if the species are the rows or columns

Details

`chao1` will return an estimate of species richness based on a vector or matrix of abundance data, while `chao2` will return an estimate of species richness based on incidence data. Note that `chao1` estimator is for abundance data only. The `chao2` estimator can be given abundance data and it will automagically convert it to incidence data, but due to the nature of the estimator, the data must contain more than one sample (ie the data must be arranged in a minimum 2 by 2 matrix).

Value

returns a value for the Chao Species Estimator for a the given data.

Note

While the function will still return a value, if all the species abundances are equal to 1 in the input to `chao1`, a warning will be raised, and the value returned will be equal to the number of species observed.

Author(s)

Matthew Vavrek

References

Chao, A. 1984. Non-parametric estimation of the number of classes in a population. Scandinavian Journal of Statistics 11: 265-270.

Chao, A. 1987. Estimating the Population Size for Capture-Recapture Data with Unequal Catchability. Biometrics 43: 783-791.

See Also

`jack1`, `bootstrap`

Examples

```
## sample vector
a<-c(0,5,1,1,2,0,0,1,0,0,8,45)
chao1(a)

## matrix format
a<-matrix(c(0,5,1,1,2,0,0,1,0,0,8,45),4,3)
chao1(a)
chao2(a)

## presence absence matrix
a<-matrix(c(0,1,1,1,1,0,0,1,0,0,1,1),4,3)
chao1(a)
chao2(a)
```

`coi` *Cohesiveness Index for Relational Clustering*

Description

Computes Cohesiveness Index for a Cluster Analysis

Usage

```
coi(mst, groups)
```

Arguments

`mst`	A minimum spanning tree matrix (binary)
`groups`	A vector with the group/cluster assignments for each sample

Value

Returns a something

Note

While the function will still return a value, if all the species abundances are equal to 1 in the input to `chao1`, a warning will be raised, and the value returned will be equal to the number of species observed.

Author(s)

Matthew Vavrek

See Also

`rclust`

Examples

```
## sample vector
a<-c(0,5,1,1,2,0,0,1,0,0,8,45)
chao1(a)

## matrix format
a<-matrix(c(0,5,1,1,2,0,0,1,0,0,8,45),4,3)
chao1(a)
chao2(a)

## presence absence matrix
a<-matrix(c(0,1,1,1,1,0,0,1,0,0,1,1),4,3)
chao1(a)
chao2(a)
```

`create.lats` *Creating a table of Latitudes and Longitudes*

Description

Create a matrix of locations with a column of latitudes and longitudes

Usage

```
create.lats(x, loc="locality", long="longitude", lat="latitude")
```

Arguments

`x`	a table arranged in columnar format, with one column indicating the locations, another the latitude and another the longitude
`loc`	the name or number of the column giving the names of the locations to be used
`long`	the name or number of the column giving the longitude of the locations
`lat`	the name or number of the column giving the latitude of the locations

Details

This function will create a location table with longitude (X) and latitude (Y) or their equivalents for every location. This function ceates a matrix in the format needed for most of the geographic functions found in the fossil package.

Value

A matrix with a column of longitude and latitude, respectively with rownames correspnding to each location

Author(s)

Matthew Vavrek

See Also

`create.matrix`

Examples

```
#to reproduce the fdata.lats dataset
data(fdata.list)
create.lats(fdata.list, loc="locality", long="longitude", lat="latitude")
```

`create.matrix` *Creating species locality matrices*

Description

Create a matrix with taxa as rows and occurrences or samples as columns

Usage

```
  create.matrix(
  x,
  tax.name="genus",
  locality="locality",
  time.col=NULL,
  time=NULL,
  abund=FALSE,
  abund.col="abundance"
)
```

Arguments

`x`	a table arranged in columnar format, with at least one column indicating name of taxa and another giving location or sample
`tax.name`	the name or number of the column giving the taxonomic names to be used (the rows of the matrix to be created)
`locality`	the name or number of the column giving the locations of the samples (the columns of the matrix to be created)
`time.col`	what is the column name or number containing the time periods; if left null, filtering for time willbe ignored
`time`	what time periods to keep for the matrix; if left null, filtering for time willbe ignored
`abund`	whether to record abundances of taxa; if left FALSE, a binary (presence/absence) matrix is created
`abund.col`	column name or number containing abundance values

Details

This is a helper function to convert large lists of data into matrices of species (rows) and locations (columns). The parameters can be adjusted to create either a binary (presence/absence) or abundance matrix. The setup of the table is largely flexible; simply input the column names or numbers containing the pertinent information. To filter data according to time, both the time column and the time period must be specified. For abundance, the default title for the abundance column is simply "abundance"; the function will not work if you have chosen to include abundances (`abund = TRUE` but the name of the abundance column is incorrect.

Value

A matrix of taxa (rows) by localities (columns).

Note

At present, the function will ignore rows where the taxon name is `NA`, `NULL`, `''` (empty character value) or `' '` (single space), as these labels typically represent an unknown taxa, which would be inappropriate to include in most analyses.

Author(s)

Matthew Vavrek

See Also

`create.lats`

Examples

```
#converting the fdata.list dataset into a matrix of species (rows)
#by samples (columns) with abundance data
data(fdata.list)
create.matrix(fdata.list, tax.name = "species", abund=TRUE)

#same data set, but now for an occurrence matrix
create.matrix(fdata.list, tax.name = "species", locality = "locality")
```

`deg.dist` *Haversine Distance Formula*

Description

Haversine formula to calculate distances between points on the earth

Usage

```
deg.dist(
long1,
lat1,
long2,
lat2
)
```

Arguments

`long1`	longitude of location 1
`lat1`	latitude of location 1
`long2`	longitude of location 2
`lat2`	latitude of location 2

Details

This function will calculate the shortest distance (portion of a Great Circle) in kilometers between two points on the Earth given their latitude and longitude.

Value

Arc distance between two points on the Earth's surface in kilometers.

Note

The distance calculated may be up to 0.2% inaccurate, as this function treats the Earth as a sphere with a circumference of 40041.47 km (mean circumference), rather than an ellipsoid like it actually is.

Author(s)

Matthew Vavrek

References

The formulas for the Haversine distance function were taken from the Dr. Math website at `http://mathforum.org/library/drmath/view/55417.html`

See Also

To calculate pairwise distances between a list of points see `earth.dist`, or to calculate an area enclosed by three points on the Earth's surface, see `earth.tri`

Examples

```
##distance between 23 degrees N 54 degrees E and 32 degrees S 67 degrees E
deg.dist(23,54,-32,67)
```

`dino.mst` *Calculate a Minimum Spanning Tree or Network*

Description

Methods for calculating a minimum spanning tree or network between a number of points given a distance matrix.

Usage

```
dino.mst(x, random.start = TRUE, random.search = TRUE)
dino.msn(x)
```

Arguments

`x`	a distance matrix for any number of points
`random.start`	If the minimum spanning tree is to start at a random point and not the first given site (default is TRUE)
`random.search`	If there is more than one shortest possible branch, should one be chosen randomly

Details

Ensure that a distance matrix is used, and not a similarity matrix, otherwise the result given will be highly incorrect.

Value

Returns a binary matrix where connections between points are denoted by a 1.

Author(s)

Yvonnick Noel, Julien Claude and Emmanuel Paradis with modifications from Matthew Vavrek

See Also

`dino.dist`

Examples

```
#minimum spanning tree for the fdata set
data(fdata.mat)
fdata.dist<-dino.dist(fdata.mat)
dino.mst<-dino.mst(fdata.dist)
```

`earth.bear` *Bearings Between Geographic Locations*

Description

Calculate the bearing in degrees clockwise from True North between any two points on the globe.

Usage

```
earth.bear(long1, lat1, long2, lat2)
```

Arguments

`long1`	Longitude of site 1
`lat1`	Latitude of site 1
`long2`	Longitude of site 2
`lat2`	Latitude of site 2

Details

Calculate the bearing in degrees clockwise from True North between any two points on the globe. Primarily designed to be used with other included geographic tools.

Value

Returns a value in degrees from True North between two geographic points.

Author(s)

Matthew Vavrek

References

Haversine formula from Math Forums: Ask Dr. Math at http://mathforum.org/dr.math/

See Also

`earth.poly`

Examples

```
earth.bear(-100, 30, 20, -40)
```

earth.dist *Calculating Geographic Distances*

Description

Create a distance matrix (lower triangle) between a list of points

Usage

```
earth.dist(lats, dist = TRUE)
```

Arguments

`lats`	a table with a longitude and latitude column respectively as the first two columns
`dist`	A logical argument whether to create a distance matrix (lower triangle) or full matrix

Details

This function will calculate the pairwise distances between all points given and return either a distance or full matrix as specified. All coordinates must be in decimal degrees.

Value

Returns a matrix of distances in kilometers between a list of longitudes and latitudes.

Note

Large datasets may take some time to process, as the number of distances to calculate is factorial in nature.

Author(s)

Matthew Vavrek, with suggestions from Anton Korobeynikovs

See Also

`deg.dist`

Examples

```
data(fdata.lats)
earth.dist(fdata.lats)
```

`earth.poly` *Calculating a Minimum Convex Polygon*

Description

Calculate a minimum convex polygon for a collection of points without knowing what points form the vertices.

Usage

```
earth.poly(lats)
```

Arguments

`lats`	a table with a longitude and latitude column respectively as the first two columns, or a SpatialPoints object with longitude/latitude

Details

This function will calculate the area of a minimum convex polygon/convex hull for a spherical surface (ie points on a globe).

Value

The function will return a list consisting of the area in $km^2$ (`$area`) and a vector with the row numbers of the vertices (`$vertices`)

Author(s)

Matthew Vavrek

See Also

`earth.tri`

Examples

```
#1/8th the surface area of the earth
a <- matrix(c(0, 0, 0, 90, 90, 0, 25, 25), 4, 2, byrow = TRUE)
earth.poly(a)
```

`earth.tri` *Calculating the Surface Area Enclosed by Three Geographic Points*

Description

Calculate the true area on a sphere enclosed by three points on the earth's surface

Usage

```
earth.tri(long1, lat1, long2, lat2, long3, lat3)
```

Arguments

`long1`	Longitude of site 1
`lat1`	Latitude of site 1
`long2`	Longitude of site 2
`lat2`	Latitude of site 2
`long3`	Longitude of site 3
`lat3`	Latitude of site 3

Details

A function to find the area enclosed by three points on the surface of the earth, given their latitudes and longitudes. This function is primarily designed to be a component of `earth.poly`, which is likely a more useful function for most applications.

Value

Returns a value in kilometers squared of the area enclosed by the three points.

Note

The distance calculated may be up to 0.2% inaccurate, as this function treats the Earth as a sphere with a circumference of 40041.47 km (mean circumference), rather than an ellipsoid like it actually is.

Author(s)

Matthew Vavrek

References

Wolfram Mathworld, `http://mathworld.wolfram.com/SphericalTriangle.html`

See Also

`earth.poly`

Examples

```
#1/8th the surface area of the earth
earth.tri(0, 0, 0, 90, 90, 0)
```

`ecol.dist` *Creating a Distance Matrix*

Description

Create a distance matrix between any number of locations

Usage

```
ecol.dist(x, method = sorenson, type = "dis")
dino.dist(x, method = sorenson, type = "dis")
```

Arguments

`x`	matrix of taxa (or equivalent data) in rows by columns of localities (or equivalent)
`method`	the distance/similarity index to compute
`type`	if the matrix is to be a distance ('dis') or similarity ('sim) matrix

Details

This will create a distance (or similarity) matrix using any of the provided indices: `sorenson`, `simpson`, `bray.curtis`, `jaccard`, `morisita.horn`, `chao.jaccard` and `chao.sorenson`. Creating a distance matrix will give a value of 1 for the most distantly related sites, while similarity index will give a value of 1 for the most similar sites.

`dino.dist` is an old name for the function, and is in the process of being deprecated.

Value

A distance matrix (lower triangle) giving the pairwise distance indices between all points.

Note

To use a user generated distance index, type the name of the function to be used for method, and the function will use that function instead. Note that the function internally provides two equal length vectors at a time to the distance calculation function.

Author(s)

Matthew Vavrek

See Also

sorenson, simpson, bray.curtis, jaccard, morisita.horn, chao.jaccard and chao.sorenson

Examples

```
##example using fdata.mat
data(fdata.mat)
ecol.dist(fdata.mat)
ecol.dist(fdata.mat,simpson,"sim")
```

`euler.rot` *Calculate the Euler Rotation of a Point*

Description

Calculate the rotation of a point on the Earth for a given Euler pole. The rotation assumes a shperical earth.

Usage

```
euler.rot(lat1, long1, rotdeg, lat2, long2)
```

Arguments

`lat1`	Euler-pole latitude
`long1`	Euler-pole longitude
`rotdeg`	Rotation about Euler-pole
`lat2`	Latitude of point to be converted
`long2`	longitude of point to be converted

Details

Locations of the Euler pole and the point to be rotated must be given in decimal degrees.

Value

Rotated latitude and longitude of the provided point in decimal degrees.

Author(s)

Matthew Vavrek

References

~put references to the literature/web site here ~

fdata *A Sample Species Abundance Dataset*

Description

A simple hypothetical data set used in many of the examples.

Value

There are 3 datasets, however 2 of them (`fdata.mat` and `fdata.lats`) derive from the first (`fdata.list`). `fdata.list` is a table with 5 columns descriing the sample site, species name, abundance, and location in latitude/longitude. `fdata.mat` is a 12 by 12 species abundance matrix (12 unique species and 12 unique samples/localities) that can be recreated from the original table of occurrences using the `create.matrix()` function; likewise, the `fdata.lats()` contains the locations of each of the samples, and can be created using the `create.lats()`

Author(s)

Matthew Vavrek

Examples

```
data(fdata.list)
```

int.chao *Internal function for chao estimators*

Description

Computes the Chao species estimator for both `chao1` and `chao2` estimators

Usage

```
int.chao(x)
```

Arguments

`x`	a vector of positive integers or zero of any length

Details

This function is typically only called internally by the functions `chao1` and `chao2`. The function has a built in bias correction, such that it will not return values of infinity or non-numbers.

Value

Estimated numer of species using the Chao estimator.

Author(s)

Matthew Vavrek

References

Chao, A. 1984. Nonparametric estimation of the number of classes in a population. Scandinavian Journal of Statistics 11: 265-270.

See Also

For the more useful implementations of the Chao estimator, see `chao1` for the abundance based estimator or `chao2` for the incidence based estimator

Examples

```
## create example data set
a<-c(4,5,1,1,2,0,0,1,3,0,8,45,23)
int.chao(a)

## a data set which would give NaN using classic (ie not bias corrected) version
a<-c(4,5,0,0,2,0,0,0,3,0,8,45,23)
int.chao(a)
```

`jack1` *First- and second-order jacknife estimators*

Description

Computes the extrapolated species richness of a population using first- or second-order jacknife stimators

Usage

```
jack1(x, taxa.row = TRUE, abund = TRUE)
jack2(x, taxa.row = TRUE, abund = TRUE)
```

Arguments

`x`	a vector, matrix or data frame of positive integers or zero of any size
`taxa.row`	whether each row of the matrix is a different taxon; if so, value is set to TRUE
`abund`	If true, data is assumed to be abundance, if false, presence absence is assumed

Details

These functions compute the first and second-order jacknife species richness estimators, respectively. Both functions will accept a vector, matrix or data frame of any size made up of positive integers and zeros. Matrices are by default treated such that each row is a different taxon and each column is a sample or locality, however if they are arranged with the taxa as columns, change the argument `taxa.row` to FALSE. If the data is abundance based, `abund` should be set to TRUE. If `abund` is set to FALSE, the data will be converted to presence/absence if not already in that format. For single vectors/columns, `taxa.row` and `abund` are ignored.

Value

The value returned is the Jackknife estimated species diversity of the dataset in question.

Author(s)

Matthew Vavrek

References

Burnham, K.P. & W.S. Overton. 1978. Estimation of the size of a closed population when capture probabilities vary among animals. Biometrika 65, 623-633.

Burnham, K.P. & W.S. Overton. 1979. Robust estimation of population size when capture probabilities vary among animals. Ecology 60, 927-936.

Heltshe, J. & Forrester, N.E. 1983 . Estimating species richness using the jackknife procedure. Biometrics 39, 1-11.

See Also

`ACE`

Examples

```
## sample vector
a<-c(0,5,1,1,2,0,0,1,0,0,8,45)
jack1(a)

## matrix format
a<-matrix(c(0,5,1,1,2,0,0,1,0,0,8,45),4,3)
jack1(a)
jack2(a)
jack2(a,abund = FALSE)

## presence absence matrix of the above abundance matrix
a<-matrix(c(0,1,1,1,1,0,0,1,0,0,1,1),4,3)
jack1(a)
jack2(a)
jack2(a, abund = FALSE)
```

`lats2Shape` *Converting a Table of Latitudes and Longitudes to a Shapefile*

Description

A helper function to convert a table of latitudes and longitudes (and associated attributes, if applicable) into a shapefile

Usage

```
lats2Shape(lats)
```

Arguments

`lats`	a table with a latitude and longitude column respectively with associated attributes

Details

The table to be converted must contain as it's first two columns the latitude (or Y) and longitude (or X) values to be converted. Any other number of columns in any format can also be attached, and will be included in the attribute table.

Value

A shapefile object which can be written to file using `write.shapefile`

Author(s)

Matthew Vavrek

See Also

`msn2Shape`

Examples

```
## Not run:
#use fdata.lats as dataset
data(fdata.lats)
shape.lats<-lats2Shape(fdata.lats)
write.shapefile(shape.lats, file='/path/to/write/lats')

## End(Not run)
```

`loc.map` *Mapping Points on a Global Map*

Description

A function to plot any number of points given their latitude and longitude respectively on a map of the world.

Usage

```
loc.map(x, ...)
```

Arguments

`x`	a table with a longitude and latitude column respectively with optional associated attributes
`...`	arguments to be passed to the `plot` call

Details

This is a helper function, which automatically zooms in and centers the map view on the input points. The `...` allow the user to adjust the usual parameters for a scatterplot outlined by `par`.

Value

Plots a map of the world focused on the locations provided.

Author(s)

Matthew Vavrek

See Also

`msn.map`

Examples

```
#plotting the fdata sample set
data(fdata.lats)
loc.map(fdata.lats)
```

`localoptima` *Function to Find Local Optimization for clustering*

Description

A function meant to be used internally be the `relational.clustering` function

Usage

```
localoptima(dist, group)
```

Arguments

`dist`	Distance matrix to be used
`group`	group designations

Details

The function takes a distance matrix and a vector with the group identifications for each sample locality (or equivalent). It is mainly meant to be used internally by the `relational.clustering` function to optimize the initial clustering and find the local (which hopefully is also the global) optimal organization, such that each member of a group is more similar to the other members in it's group (on average) than to any other groups.

Value

Arc distance between two points on the Earth's surface in kilometers.

Note

The distance calculated may be up to 0.2% inaccurate, as this function treats the Earth as a sphere with a circumference of 40003 km, rather than an ellipsoid like it actually is.

Author(s)

Matthew Vavrek

See Also

To calculate pairwise distances between a list of points see `earth.dist`, or to calculate an area enclosed by three points on the Earth's surface, see `earth.tri`

Examples

```
##distance between 23 degrees N 54 degrees E and 32 degrees S 67 degrees E
deg.dist(23,54,-32,67)
```

msn.map *Mapping a Minimum Spanning Tree*

Description

Creating a quick and focused map using a world map for gegraphically referenced visualization within R of a minimum spanning tree or network.

Usage

```
msn.map(msn, lat, ...)
```

Arguments

`msn`	minimum spanning tree or network to be used
`lat`	the lats
`...`	arguments to be passed to `plot`

Details

This is a helper function for quick visualization of georeferenced minimum spanning trees, and is not meant for creating figure quality images due to lack of fine control over many functions

Value

Returns a map of the globe, focused in on any set of georeferenced localities.

Author(s)

Matthew Vavrek

See Also

`dino.msn`

Examples

```
##add examples
```

msn2Shape — *Convert a Minimum spanning Network or Tree to Shapefile*

Description

A helper function to convert a minimum spanning tree or network into shapefile format.

Usage

```
msn2Shape(msn, lats, dist = NULL)
```

Arguments

`msn`	a minimum spanning tree or network (binary matrix)
`lats`	a matrix or data frame with the latitude and longitude of the sites as the first two columns respectively
`dist`	Optional argument to include distance values in final output; if wanted, a distance matrix (lower triangle) with the localities in the same order as in the MSN are required

Details

This function will take a minimum spanning tree or network object, along with the georeferenced locations of the sites, and convert it into a shapefile for use with GIS. The `msn` argument requires a minimu spanning tree or network object, and the lat argument requires some form of location for each of the points, typically a matrix with latitude and longitude columns respectively.

Value

A shapefile which can be output using the `write.shapefile` function for use with a GIS program.

Author(s)

Matthew Vavrek

See Also

`lats2Shape` for a function to convert a lat/long table to a shapefile

Examples

```
## Not run:
#import both fdata.lats and fdata.mat
data(fdata.lats)
data(fdata.mat)
fdata.dist<-dino.dist(fdata.mat)
fdata.mst<-dino.mst(fdata.dist)
shape.mst<-msn2Shape(fdata.mst, fdata.lats)
```

```
write.shapefile(shape.mst, file='/path/to/write/mst')

## End(Not run)
```

`mstlines` *Display a Minimum Spanning Tree or Network*

Description

a method of displaying a Minimum Spanning Tree/Network over a given set of points

Usage

```
mstlines(mst, x, y = NULL, pts.names = NULL, ...)
```

Arguments

`mst`	a minimum spanning tree or network object
`x`	either a table with the first two columns that of the x and y coordinates respectively, or simply that of the x coordinate
`y`	an optional argument if the y coordinates were not given in argument x
`pts.names`	If there is more than one shortest possible branch, should one be chosen randomly
`...`	arguments to be passed to lines()

Details

A function to plot the lines of a minimum spanning tree/forest on a plot; works as a frontend for `lines`.

Author(s)

Matthew Vavrek

See Also

`dino.dist`

Examples

```
#plot with overlain MST for fdata dataset
data(fdata.lats)
data(fdata.mat)
fdata.dist<-dino.dist(fdata.mat)
fdata.mst<-dino.mst(fdata.dist)
plot(coordinates(fdata.lats))
mstlines(fdata.mst, coordinates(fdata.lats))
```

`new.lat.long` *Find a New Latitude and Longitude*

Description

Find a new location using an original location (latitude and longitude) along with a bearing and distance

Usage

```
new.lat.long(long, lat, bearing, distance)
```

Arguments

`long`	original longitude
`lat`	original latitude
`bearing`	bearing from original point to new location, degrees from North
`distance`	distance to location

Value

a vector of length 2 with the new latitude and longitude respectively

Author(s)

Matthew Vavrek

See Also

`deg.dist`, `earth.bear`

Examples

```
#Travel from 0,0 to a new location at a bearing of 45 degrees
#from North (clockwise) and 1000 km away
new.lat.long(long = 0, lat = 0, bearing = 45, distance = 1000)
```

nmds.mst *Creating NMDS plots with overlain Minimum Spanning Trees*

Description

This is a helper function which will plot an NMDS with an overlain MST

Usage

```
nmds.mst(nmds, mst, ...)
```

Arguments

nmds	an NMDS created using the ecodist program
mst	a minimum spanning tree or network (binary matrix)
...	arguments to be passed to the `plot` function

Details

At the moment, the function requires an NMDS created using the ecodist program, howver the minimum spanning tree can be any one which creates a binary matrix showing connections (ie `dino.mst`).

Value

Plots a non-metric multidimensional scaling plot with an overlain minimum spanning tree showing connections between the points.

Author(s)

Matthew Vavrek

See Also

`dino.msn`, `dino.mst`

Examples

```
## Not run:
#use fdata.mat as dataset, and use the \code{ecodist} package for the \code{nmds()} function
data(fdata.mat)
z <- ecol.dist(fdata.mat)
a <- dino.msn(z)
b <- nmds(z)
nmds.mst(b, a)

## End(Not run)
```

`rand.index` *Rand Index and Adjusted Rand Index*

Description

Measures to compare the similarity of two clustering outcomes

Usage

```
rand.index(group1, group2)
adj.rand.index(group1, group2)
```

Arguments

`group1`	first cluster identity matrix
`group2`	second cluster identity matrix

Details

This function calculates the Rand Index for two different clustering outcomes. The Rand Index gives a value between 0 and 1, where 1 means the two clustering outcomes match identicaly.

The Adjusted Rand Index rescales the index, taking into account that random chance will cause some objects to occupy the same clusters, so the Rand Index will never actually be zero.

Value

a single value between 0 and 1

Author(s)

Matthew Vavrek

References

Rand, W.M. 1971. Objective criteria for the evaluation of clustering methods. Journal of the American Statistical Association 66: 846–850.

Hubert, L. and Arabie, P. 1985. Comparing partitions. Journal of Classification. 2: 193–218.

See Also

To cluster the data, use the function `rclust`

Examples

```
#create a hypothetical clustering outcome with 2 distinct clusters
g1 <- sample(1:2, size=10, replace=TRUE)
g2 <- sample(1:3, size=10, replace=TRUE)
rand.index(g1, g2)
```

`rclust` *Relational Clustering*

Description

A method to cluster a number of samples using a relational (dissimila

Usage

```
rclust(dist, clusters = 2, runs = 10, counter = FALSE)
```

Arguments

`dist`	pairwise distance matrix
`clusters`	final number of clusters
`runs`	number of randomizations to run
`counter`	use counter to display current run number

Details

This function will return a cluster identity vector. The number of clusters requested must be 2 or greater, but 1/2 or less than the total number of samples, as the function requires at least 2 samples per cluster.

Value

The vector returned will be the same length as the number of samples provided in the original `dist` matrix. The samples will have been clustered into the same number of clusters as defined in `clusters`

Author(s)

Matthew Vavrek

See Also

`rclust.dist`, `rclust.null`, `rclust.weights` and `coi`

Examples

```
#a null solution for the fdata example data set
data(fdata.mat)
fd.dist <- dino.dist(fdata.mat)
rclust(fd.dist, clusters = 2, runs = 10)
```

rclust.dist *Relational Clustering*

Description

Provides a distance matrix intra- and inter-group average distances based on a clustering solution and a dissimilarity matrix.

Usage

```
rclust.dist(groups, dist)
```

Arguments

`groups`	cluster identity vector
`dist`	original pairwise distance matrix

Details

This function calculates a distance matrix for each cluster, giving the average within group pairwise distance and the average between group pairwise distance.

Value

A matrix c by c in size, where c is the number of clusters

Author(s)

Matthew Vavrek

See Also

To cluster the data, use the function `rclust`; see as well `rclust.weights`, `rclust.null`

Examples

```
#a null solution for the fdata example data set
data(fdata.mat)
fd.dist <- dino.dist(fdata.mat)
fd.clust <- rclust(fd.dist, 2)
rclust.dist(fd.clust, fd.dist)
```

`rclust.null` *Relational Clustering*

Description

A Monte Carlo method for calculating a null/random clustering solution based on the type and arrangement of a known clustering solution.

Usage

```
rclust.null(groups, dist)
```

Arguments

`groups`	cluster identity vector
`dist`	original pairwise distance matrix

Details

This function calculates a random/null clustering solution based on a given solution. It resamples the data and reassigns samples to groups, keeping the same group sizes.

Value

A matrix equal in rows to the number of clusters originally given, with two columns for mean within group distance and standard deviation respectively.

Author(s)

Matthew Vavrek

See Also

To cluster the data, use the function `rclust`; see as well `rclust.weights`, `rclust.dist`

Examples

```
#a null solution for the fdata example data set
data(fdata.mat)
fd.dist <- dino.dist(fdata.mat)
fd.clust <- rclust(fd.dist, 2)
rclust.null(fd.clust, fd.dist)
```

`rclust.weights` *Relational Clustering*

Description

A method to cluster a number of samples using a relational (dissimila

Usage

```
rclust.weights(groups, dist)
```

Arguments

`groups`	cluster identity vector
`dist`	original pairwise distance matrix

Details

This function creates an n by c sized matrix, where n is the number of samples and c is the number of groups, of the average distances for each sample from itself to all the members of another group (c_i).

Value

A matrix of of size n (samples) by c (groups).

Author(s)

Matthew Vavrek

See Also

To cluster the data, use the function `rclust`; see as well `rclust.dist`, `rclust.null`

Examples

```
data(fdata.mat)
fd.dist <- dino.dist(fdata.mat)
fd.clust <- rclust(fd.dist, 2)
rclust.weights(fd.clust, fd.dist)
```

relational.clustering *Relational Clustering*

Description

A method to cluter a number of samples using a relational (dissimila

Usage

```
relational.clustering(dist, clusters = 2)
```

Arguments

`dist`	pairwise distance matrix
`clusters`	number of clusters required

Details

This function will calculate the shortest distance (portion of a Great Circle) in kilometers between two points on the Earth given their latitude and longitude.

Value

Arc distance between two points on the Earth's surface in kilometers.

Note

The distance calculated may be up to 0.2% inaccurate, as this function treats the Earth as a sphere with a circumference of 40003 km, rather than an ellipsoid like it actually is.

Author(s)

Matthew Vavrek

References

The formulas for the Haversine distance function were taken from the Dr. Math website at `http://mathforum.org/library/drmath/view/55417.html`

See Also

To calculate pairwise distances between a list of points see `earth.dist`, or to calculate an area enclosed by three points on the Earth's surface, see `earth.tri`

Examples

```
##distance between 23 degrees N 54 degrees E and 32 degrees S 67 degrees E
deg.dist(23,54,-32,67)
```

`sac` *Calculate Species Area Curves*

Description

Calculating a species area curve for a set of georeferenced localities

Usage

```
sac(lats, spp)
```

Arguments

`lats`	a table with a longitude and latitude column respectively as the first two columns, or a SpatialPoints object with longitude/latitude
`spp`	A matrix/data frame of species (rows) by samples/localities (columns)

Details

This will take a set of geographic coordinates along with a table of species by localities and return a list consisting of a matrix (`$areavsspp`) with a column of total area and of total species present, and a vector (`ranks`) with the order the samples were added in. The area is calculated by starting with the most central point, and adding those points closest to it, calculating a minimum spanning polygon as each new site is added, until all points are used.

Value

Returns a list of a matrix with columns of total area and total species recorded respectively and a vector of sample orders.

Author(s)

Matthew Vavrek

See Also

`earth.dist`, `earth.poly`

Examples

```
#fdata species/area relationship
data(fdata.lats)
data(fdata.mat)
a<-sac(fdata.lats, fdata.mat)
plot(log(a$areavsspp))
```

sim.occ *Simulated Species Occurrence data*

Description

A function to simulate a species occurrence data set

Usage

```
sim.occ(total.species = 100, endemics = 0.1, regions = 3, locs = 30, avg.abund = 1)
```

Arguments

`total.species`	The total number of species in the region (i.e. the number of rows in the result matrix)
`endemics`	The proportion of endemic species for the entire region
`regions`	The number of areas of endemicity
`locs`	The number of samples/locatlities per region of endemicity
`avg.abund`	The 'average' abundance of a species for any given sample

Details

The function creates a matrix of \$c\$ rows of species (given by `total.species`) with \$n\$ number of sample columns (where \$n\$ equals \$regions*locs\$). The given abundance of any species at a given sample is determined by a log normal distribution, with each species being randomly assigned a value from `rnorm()`. The number of endemics for any given region is equal to \$total.species*endemics/regions\$. An endemic is conseidered to only occur within a given region, and all other non-ndemic species are considered to be 'cosmopolitan' and can occur in any region. The `avg.abund` value affects how many species are recovered at a given site, and for any given run there are typically species that are not present in the sample but are present in the region.

Value

Returns a matrix of simulated species abundances per locality.

Author(s)

Matthew Vavrek

See Also

`ecol.dist`

Examples

```
## create a dataset with 2 regions and 5 samples per region
sim.occ(regions=2, locs=5)
```

`similarity` *Similarity/Dissimilarity Indices*

Description

Functions to calculate the ecological distance between two groups

Usage

```
braun.blanquet(x, y)
bray.curtis(x, y)
euclidean(x,y)
kulczynski(x,y)
jaccard(x, y)
manhattan(x, y)
morisita.horn(x, y)
ochiai(x, y)
simpson(x, y)
sorenson(x, y)
```

Arguments

`x`	species from group A
`y`	species from group B

Details

You must provide two separate vectors, with species arranged in the same order, from area A and B. If species are present in one site but not the other, these must be recorded for both sites; the site where they are not found should be coded as a zero. For details on each index, please consult the references.

Value

Returns the similarity index for the two sites in question.

Author(s)

Matthew Vavrek

References

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

Magurran, A. E. 2004. Measuring Biological Diversity. Oxford, Blackwell.

See Also

dino.dist

Examples

```
##Species counts from two different locations
a <- c(1,0,4,3,5,0,0,7)
b <- c(2,1,3,0,0,1,0,6)
bray.curtis(a,b)
jaccard(a,b)
simpson(a,b)
sorenson(a,b)
morisita.horn(a,b)
```

spp.est *Estimating Species Diversity*

Description

Estimate the diversity of a sample(s) using a number of species diversity estimators.

Usage

```
spp.est(x, rand = 10, abund = TRUE, counter = FALSE, max.est = 'all')
```

Arguments

x	A vector, matrix or data frame with species as rows and locations/samples as columns
rand	The number of times to run the internal randomizations; default is set to 10
abund	If the data is abundance or presence/absence; default is set to TRUE for abundance
counter	Whether or not to provide a running total of progress of randomizations
max.est	The value to go up to for the analysis; default is set to the same as the total number of samples

Details

This function will accept a vector, matrix or data frame of species by samples and return a large matrix with various species estimation values.

Value

Returns a table with the following column names if `abund=TRUE`:

`N.obs`	Total sample size
`S.obs`	Number of observed species
`S.obs(+95%)`	95% upper confidence interval
`S.obs(-95%)`	95% lower confidence interval
`Chao1`	Chao Species Estimation
`Chao1(upper)`	95% upper confidence interval
`Chao1(lower)`	95% lower confidence interval
`ACE`	Abundance-based Coverage Estimator
`ACE(upper)`	95% upper confidence interval
`ACE(lower)`	95% lower confidence interval
`Jack1`	First Order Jacknife Estimator
`Jack1(upper)`	95% upper confidence interval
`Jack1(lower)`	95% lower confidence interval

Returns a table with the following column names if `abund=FALSE`:

`N.obs`	Total sample size
`S.obs`	Number of observed species
`S.obs(+95%)`	95% upper confidence interval
`S.obs(-95%)`	95% lower confidence interval
`Chao2`	Chao Species Estimation
`Chao2(upper)`	95% upper confidence interval
`Chao2(lower)`	95% lower confidence interval
`ICE`	Incidence-based Coverage Estimator
`ICE(upper)`	95% upper confidence interval
`ICE(lower)`	95% lower confidence interval
`Jack1`	First Order Jacknife Estimator
`Jack1(upper)`	95% upper confidence interval
`Jack1(lower)`	95% lower confidence interval

Note

This function can be very long to run due to its iterative nature. The randomizations are initially set to 10 so the process will run relatively quickly, but a low value for randomizations will not give nicely smoothed curves.

Also, in some cases due to the nature of some of the functions, they provide no answer, such as is common with the Chao standard deviation. In this case, the Chao upper and lower bounds are simply 95% confidence intervals based on the actual Chao estimator.

Author(s)

Matthew Vavrek

References

The original idea for a program similar to this came from the extremely useful EstimateS program by Robert K. Colwell

Colwell, R.K. 2010. EstimateS: Statistical estimation of species richness and shared species from samples. Version 8.2. User's Guide and application published at: http://purl.oclc.org/estimates.

See Also

`chao1`, `jack1`, `bootstrap`

Examples

```
#abundance example with sample data set
data(fdata.mat)
spp.est(fdata.mat, abund = TRUE, counter = FALSE)

#occurrence example with sample data set
data(fdata.mat)
spp.est(fdata.mat, abund = FALSE, counter = FALSE)
```

`tri.ineq` *Testing for the Triangle Inequality*

Description

Determines if a distance matrix obeys the triangle inequality

Usage

```
tri.ineq(dist)
```

Arguments

`dist`	A distance matrix

Details

Tests if a distance matrix respects the triangle inequality. Often with non-monotonic distance measures and complex data a situation can arise where the triangle inequality (where no single side of a triangle is greater in length than the sum of the other two sides) is not respected.

Value

Returns a TRUE if the inequality is respected, and a FALSE if there is any situation where the triangle inequality is not respected.

Author(s)

Matthew Vavrek

See Also

`ecol.dist`

Examples

```
## sample distance matrix with an impossible triangle
a<-matrix(0.2, 4,4)
a[4,2]<-0.8
a<-as.dist(a)
tri.ineq(a)
```

Index