Package 'phylin'

July 23, 2025

Type Package

Title Spatial Interpolation of Genetic Data

Version 2.0.2

Date 2019-11-19

Author Pedro Tarroso, Guillermo Velo-Anton, Silvia Carvalho

Maintainer Pedro Tarroso <ptarroso@cibio.up.pt>

Suggests geometry,raster,gdistance

Description The spatial interpolation of genetic distances between samples is based on a modified kriging method that accepts a genetic distance matrix and generates a map of probability of lineage presence. This package also offers tools to generate a map of potential contact zones between groups with user-defined thresholds in the tree to account for old and recent divergence. Additionally, it has functions for IDW interpolation using genetic data and midpoints.

License GPL (>= 2)

URL https://www.r-project.org, https://github.com/ptarroso/phylin

NeedsCompilation no

Repository CRAN

Date/Publication 2019-12-12 14:40:09 UTC

Contents

`phylin-package` *Phylogenetic Landscape Interpolation.*

Description

This package provides functions for the spatial interpolation of genetic distances between samples. The interpolation is based on a modified kriging method that accepts a genetic distance matrix and generates a map of probability of lineage presence. This package also offers tools to generate a map of potential contact zones between groups with user-defined thresholds in the tree to account for old and recent divergence. Additionally, it has functions for IDW interpolation using genetic data and midpoints.

Details

Package:	phylin
Type:	Package
Version:	2.0
Date:	2016-04-27
License:	GPL-2

The kriging algorithm uses a model fitted to the semi-variogram to weight the values of the samples. Here the variogram was modified to fit a model with pairwise comparison between genetic and real distances, describing the spatial dependence in the genetic distance between samples. A map for the lineage can be generated using only a vector that define if each point belong to the a lineage or not.

Since version 2.0, the kriging interpolation can be performed taking into consideration a cost distance instead of simple geographical distances between points. This can help in cases where a

landscape resistance explains better the genetic distances between samples than the geographic distances alone.

The IDW can be used to interpolate the genetic distance of each sample against the others, or to interpolate genetic diverge at midpoints between samples. The interpolated value at certain location is obtained by weighting with the distances to the avaialable samples. Similarly to kriging, these distance can be based on a cost distance calculation.

Author(s)

Pedro Tarroso, Guillermo Velo-Anton, Silvia Carvalho

Maintainer: Pedro Tarroso <ptarroso@cibio.up.pt>

References

Tarroso, P., Velo-Anton, G., & Carvalho, S.B. (2015). PHYLIN: an R package for phylogeographic interpolation. Molecular Ecology Resources, 15(2), 349-357.

Tarroso, P., Carvalho, S.B. & Velo-Anton, G. (2019). PHYLIN 2.0: Extending the phylogeographic interpolation method to include uncertainty and user-defined distance metrics. Molecular Ecology Resources, in press.

Examples

```
## See examples for the included functions.
```

---

d.gen *Genetic distance matrix between vipers and lineages.*

Description

This is a matrix of genetic distances between the *Vipera latastei* samples. The values are cophenetic distances generated from the phylogenetic tree.

Usage

```
data(d.gen)
```

Format

'd.gen' is a matrix with 58 rows and columns. Columns and rows are organized with the same order found in the 'vipers' dataset.

References

Velo-Anton G., Godinho R., Harris D. J. *et al.* (2012) Deep evolutionary lineages in a Western Mediterranean snake (*Vipera latastei/monticola* group) and high genetic structuring in Southern Iberian populations. *Molecular phylogenetics and evolution*, **65**, 965–973.

Examples

```
data(d.gen)
hc <- hclust(as.dist(d.gen))
plot(hc, hang = -1, main="Vipers genetic distance tree",
     xlab="Samples", cex=0.7)
```

`extract.val` *Extact pairwise values from a matrix in a specified order.*

Description

This function extacts pairwise values from a matrix in a specified order in a user provided table.

Usage

```
extract.val(m, samples)
```

Arguments

`m`	Matrix with values to extract.
`samples`	Data frame with columns indicating pairs of samples to extract values. Names must correspond to column and row names in the matrix.

Details

This function extracts the values from a matrix in the same pairs of populations/samples given in a table. It is useful for merging data from a distance matrix of samples and the midpoints between samples (in conjuction with `midpoints` function).

Value

Returns a vector containing the values from the matrix m in the order given in samples.

Author(s)

Pedro Tarroso <ptarroso@cibio.up.pt>

See Also

`dist` `d.gen` `midpoints` `idw`

Examples

```
data(vipers)
data(d.gen)

# calculate midpoints
mp <- midpoints(vipers[,1:2])

# extract values from d.gen. Columns 1 and 2 of mp have the information
# about source and target samples.
pair.data <- extract.val(d.gen, mp[,1:2])

# it is easier to view in a plot:
plot(vipers[,1:2], pch=vipers[,3], main="Midpoints between samples",
     xlab="Longitude", ylab="Latitude")
#trace all connecting lines between samples
sps <- rownames(vipers)
for (i in 1:nrow(mp))
{
    sp <- mp[i, 1:2] #source an target samples
    mask <- c(which(sps == sp[,1]), which(sps == sp[,2]))
    lines(vipers$x[mask], vipers$y[mask], lty=2, col='lightgrey')
}

#midpoints with genetic distance acentuated
points(mp[,3:4], col='red', pch=16, cex=pair.data*15+0.5)
```

`gen.variogram` *Semi-variogram with the genetic distance matrix*

Description

Computes the semi-variance with the real and genetic distances, and with user defined lag parameters.

Usage

```
gen.variogram(x, y, lag = quantile(as.matrix(x), 0.05), tol=lag/2, lmax = NA,
              bootstraps = 999, verbose = FALSE)
```

Arguments

`x`	Real distances matrix.
`y`	Single genetic distances matrix or list of genetic distances matrices.
`lag`	Real distance corresponding to the desired 'lag' interval. This is used to calculate lag centers from 0 to 'lmax'.
`tol`	Tolerance for the lag center to search for pairs ('lag'-'tol', 'lag'+'tol').

`lmax`	Maximum distance for lag centers. Pairs with distances higher than 'lmax' are not included in the calcualtion of the semi-variance. If 'lmax' is NA (default) then is used the maximum distance between samples.
`bootstraps`	This is the number of bootstraps used to calculate 95% confidence interval for the median, when multiple genetic distances are given. With a single genetic distance, this parameter is ignored.
`verbose`	Boolean for verbosity. When TRUE and with multiple genetic distance matrices, a log of error evolution is printed.

Details

This function produces a table with real lag centers and semi-variance. The formula to calculate semi-variance, $\gamma(h)$, is:

$$\gamma(h) = \frac{1}{2n(h)} \sum_{i=1}^{n} [z(x_i + h) - z(x_i)]^2$$

where $n(h)$ is the number of pairs with the lag distance h between them, and z is the value of the sample x at the the location i. The difference between sample $z(x_i + h)$ and sample $z(x_i)$ is assumed to correspond to their genetic distance.

Multiple genetic distance matrices can be used. In this case, a variogram is computed for each genetic distance and the results summarised by the median and a 95% confidence interval calculated with bootstraps.

Value

Returns a 'gv' object with the input data, lag centers and semi-variance.

Note

It is assumed that the order of samples in x corresponds to the same in y.

Author(s)

Pedro Tarroso <ptarroso@cibio.up.pt>

References

Fortin, M. -J. and Dale, M. (2006) *Spatial Analysis: A guide for Ecologists*. Cambridge: Cambridge University Press.

Isaaks, E. H. and Srivastava, R. M. (1989) *An Introduction to applied geostatistics*. New York: Oxford University Press.

Legendre, P. and Legendre, L. (1998) *Numerical ecology*. 2nd english edition. Amesterdam: Elsevier

See Also

`plot.gv` `predict.gv` `gv.model`

Examples

```
data(vipers)
data(d.gen)

# create a distance matrix between samples
r.dist <- dist(vipers[,1:2])

# variogram table with semi-variance and lag centers
gv <- gen.variogram(r.dist, d.gen)

# plot variogram
plot(gv)

# fit a new variogram with different lag
gv2 <- gen.variogram(r.dist, d.gen, lag=0.2)
plot(gv2)
```

`geo.dist` *Geographical distance matrix for samples and interpolation locations.*

Description

Calculates a geographical euclidean distances matrix based on a list of coordinates for samples and for interpolation locations.

Usage

```
geo.dist(from, to)
```

Arguments

`from`	Data frame with coordinates for source locations. Should have two columns (longitude and latitude).
`to`	Data frame with coordinates for destination locations to where distances are calculated. Should have two columns (longitude and latitude).

Value

Return the matrix of euclidean distances between source locations ('from') to destination ('to') coordinates. The resulting matrix has source locations in rows and destination in columns and respective names are given based on the row names of the 'from' and 'to' data frames.

Author(s)

Pedro Tarroso <ptarroso@cibio.up.pt>

Examples

```
data(vipers)

# create a grid of the sampled area
grid <- expand.grid(x=seq(-10,10,0.5), y=seq(30, 50, 0.5))

rd <- geo.dist(vipers[,1:2], grid)
```

`grid` *Grid centroids for the Iberian Peninsula.*

Description

This is a list of coordinates representing the interpolation area in the Iberian Peninsula with a resolution of 0.09 degrees (~10km).

Usage

```
data(grid)
```

Format

The data format is a table with two columns (longitude, and latitude) and 7955 rows (pixels).

Examples

```
data(grid)
plot(grid, cex=0.5, asp=1, main="Grid of pixels for interpolation",
     xlab="Longitude", ylab="Latitude")
```

`grid.image` *Simple plot of interpolated grid.*

Description

Plots the interpolated grid.

Usage

```
grid.image(intpl, grid, breaks=10, ic=1, colFUN=heat.colors,
           main=colnames(intpl)[ic], xlab=colnames(grid)[1],
           ylab=colnames(grid)[2], sclab=NA, ...)
```

Arguments

intpl	A matrix or vector with interpolation results.
grid	A table containing longitude and latitude of interpolated locations.
breaks	Number of breaks in the scale.
ic	Column index or name from 'intpl' table to show. Defaults to the first column. Can be used to plot standard deviation or any other column. This value is ignored of 'intpl' is a vector.
colFUN	Function to process colors. Can be any of R base color functions (e.g. rainbow, terrain.colors, etc) or user defined function.
main	Main title.
xlab	X axis label. Defaults to name of the first 'grid' column.
ylab	Y axis label. Defaults to name of the secont 'grid' column.
sclab	Scale label to plot under the scale bar.
...	Futher arguments to be passed to par. Most used is 'cex' to control the font size.

Details

This function may be used to produce a simple plot of the interpolated grid. It has some customizable features and it plots a scale bar of the Z values shown.

Note

Does not work with multiple plots (e.g. with 'layout').

Author(s)

Pedro Tarroso <ptarroso@cibio.up.pt>

See Also

image krig idw

Examples

```
data(vipers)
data(d.gen)

# create a grid of the sampled area for inteprolation
grid <- expand.grid(x=seq(-9.5,3,0.25), y=seq(36, 43.75, 0.25))

# create a distance matrix between samples
r.dist <- dist(vipers[,1:2])

# fit a model with defaults (shperical model) and estimation of range
gv <- gen.variogram(r.dist, d.gen, 0.25)
gv <- gv.model(gv)
```

```
# interpolation of the distances to first sample with ordinary kriging
int.krig <- krig(d.gen[,1], vipers[,1:2], grid, gv)

#plot the interpolation results
grid.image(int.krig, grid, main='Krigging Interpolation',
           xlab='Longitude',ylab = 'Latitude',
           sclab=paste('Genetic distance to sample',
           colnames(d.gen)[1]))

# User can add extra elements to the main plot.
points(vipers[,1:2], cex=d.gen[,1]*15+0.2)
```

`gv.model` *Fit a model to the semi-variogram.*

Description

Fits a model to a semi-variogram contructed with `gen.variogram`. Parameters for each model are estimated by nonlinear least squares.

Usage

```
gv.model(gv, model='spherical', sill = NA, range=NA, nugget = 0,
         ctrl=nls.control())
```

Arguments

`gv`	'gv' object from the `gen.variogram` function.
`model`	Model to fit the data. Available models are spherical (default), gaussian, exponential or linear. See details.
`sill`	The heigth (semi-variance) of the model when it stabilizes. Defaults to NA.
`range`	The length (real distance) where stabilization occurs. Defaults to NA.
`nugget`	Intercept in the y-axis. Defaults to 0.
`ctrl`	Nls control object for the fitting procedure. (check ?nls.control for more details).

Details

This function fits a model to the data, either by estimating the model parameters using nonlinear least squares or by user provided values. Notice that parameters with NA will be estimated by the nls function and parameters with values given are not estimated (by default, the nugget is not estimated and set to zero). The variogram model can be plotted using 'plot' function or used to predict to a new set of values using 'predict'. It is used to define weights for the krigging interpolation.

The parameters of the semi-variogram model (γ) are the distance (h), range (a), sill ($c = c_0 + c_1$; where c_1 is the partial sill), and nugget (c_0). The models available are:

1. gaussian:

$$\gamma(h) = c_0 + c_1 \left(1 - exp\left(-3\frac{h^2}{a^2}\right)\right)$$

2. exponential:

$$\gamma(h) = c_0 + c_1 \left(1 - exp\left(-\frac{3h}{a}\right)\right)$$

3. spherical:

$$\gamma(h) = \begin{cases} c_0 + c_1 \left(\frac{3}{2}\frac{h}{a} - \frac{1}{2}(\frac{h}{a})^3\right) & 0 < h \leq a \\ c & h > a \end{cases}$$

4. pentaspherical:

$$\gamma(h) = \begin{cases} c_0 + c_1 \left(\frac{15}{8}\frac{h}{a} - \frac{5}{4}(\frac{h}{a})^3 + \frac{3}{8}(\frac{h}{a})^5\right) & 0 < h \leq a \\ c & h > a \end{cases}$$

5. cubic:

$$\gamma(h) = \begin{cases} c_0 + c_1 \left(7(\frac{h}{a})^2 - \frac{35}{4}(\frac{h}{a})^3 + \frac{7}{2}(\frac{h}{a})^5 - \frac{3}{4}(\frac{h}{a})^7\right) & 0 < h \leq a \\ c & h > a \end{cases}$$

6. linear:

$$\gamma(h) = \begin{cases} c_0 + \left(\frac{c_1}{a}h\right) & 0 < h \leq a \\ c & h > a \end{cases}$$

Value

Returns a 'gv' object with the model, input data, and parameter values.

Author(s)

Pedro Tarroso <ptarroso@cibio.up.pt>

References

Fortin, M. -J. and Dale, M. (2006) *Spatial Analysis: A guide for Ecologists*. Cambridge: Cambridge University Press.

Isaaks, E. H. and Srivastava, R. M. (1989) *An Introduction to applied geostatistics*. New York: Oxford University Press.

Legendre, P. and Legendre, L. (1998) *Numerical ecology*. 2nd english edition. Amesterdam: Elsevier

See Also

`plot.gv` `predict.gv`

Examples

```
data(vipers)
data(d.gen)

# create a distance matrix between samples
r.dist <- dist(vipers[,1:2])

# fit a variogram with defaults (shperical model) and estimation of range
gv <- gen.variogram(r.dist, d.gen)
gv <- gv.model(gv)

# plot variogram
plot(gv)

# fit a new variogram with linear with sill model and range 8
gv2 <- gv.model(gv, model='linear', range=8)
plot(gv2)
```

---

idw *Inverse Distance Weighting interpolation*

Description

This function interpolates a list of samples with location and a value to a table of coordinates, that generally represent a spatial grid. The interpolation is based on inverse distance weighting algoritm with three different methods available for weight calculation.

Usage

```
idw(values, coords, grid, method = "Shepard", p = 2, R = 2, N = 15,
    distFUN = geo.dist, ...)
```

Arguments

values	A table of points to be interpolated. Table must contain x and y locations, and a column of values to be interpolated.
coords	A table wit x and y coordinates of the samples.
grid	Coordinates of locations to interpolate. It is assumed to be in the same order as 'values' table.
method	Method to calculate weights for idw. Should be "Shepard" (default), "Modified", "Neighbours", or distinctive abreviations of each. See details section for additional help on each method.
p	The power to use in weight calculation.
R	Radius to use with Modified Shepard method.
N	Maximum number of neighbours to use with Shepard with neighbours.

distFUN	Distance function used to calculate distances between locations. The default is 'geo.dist' which calculates simple euclidean distances between the locations. This function must have a 'from' and a 'to' arguments to specify, respectively, the source and destination localities.
...	Other arguments to be passed to distFUN.

Details

The IDW interpolation algorithm is commonly used to interpolate genetic data over a spatial grid. This function provides a simple interface to interpolate such data with three methods:

1. *Shepard*: weights are the inverse of the distance between the interpolation location x and the sample points x_i, raised to the power p

$$w(x) = \frac{1}{d(x, x_i)^p}$$

2. *Modified Shepard*: distances are weighted with a search radius r to calculate the interpolation weights

$$w(x) = \left(\frac{r - d(x, x_i)}{r.d(x, xi)} \right)^p$$

3. *Shepard with neighbours*: A maximum ammount of N neighbours is allowed to the weight calculation following Shepard method.

Value

It return a vector for each row of the 'coords' table with the respective interpolated value.

Author(s)

Pedro Tarroso <ptarroso@cibio.up.pt>

References

Fortin, M. -J. and Dale, M. (2006) *Spatial Analysis: A guide for Ecologists*. Cambridge: Cambridge University Press.

Isaaks, E. H. and Srivastava, R. M. (1989) *An Introduction to applied geostatistics*. New York: Oxford University Press.

Legendre, P. and Legendre, L. (1998) *Numerical ecology*. 2nd english edition. Amesterdam: Elsevier

Vandergast, A. G.,Hathaway, S. A., Fisher, R. N., Boys, J., Bohonak, A. J., (2008) Are hotspots evolutionary potential adequately protected in southern California? *Biological Conservation*, **141**, 1648-1664.

See Also

intgen.idw krig

Examples

```
data(vipers)
data(d.gen)
data(grid)

# interpolate and plot the genetic distances for sample s2 in the d.gen
int <- idw(d.gen[,2], vipers[,1:2], grid)

grid.image(int, grid, main='IDW interpolation', xlab='Longitude',
           ylab='Latitude', sclab="Genetic distance to sample s2")

points(vipers[,1:2], cex=d.gen[,2]*15+0.2)

# change idw power (i.e. points will have a larger influence in the
# surroundings)
int <- idw(d.gen[,2], vipers[,1:2], grid, p=5)

result <- data.frame(grid, int)
grid.image(int, grid, main='IDW interpolation', xlab='Longitude',
           ylab='Latitude', sclab="Genetic distance to sample s2")

points(vipers[,1:2], cex=d.gen[,2]*15+0.2)


# change idw method to "Modified Shepard" and define a maximum
# neighbour distance
int <- idw(d.gen[,2], vipers[,1:2], grid, 'Modified', R=10)

grid.image(int, grid, main='IDW interpolation', xlab='Longitude',
           ylab='Latitude', sclab="Genetic distance to sample s2")

points(vipers[,1:2], cex=d.gen[,2]*15+0.2)

#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#
#        Example following methods in Vandergast et al. 2008        #
#            Fit a linear model and recover the residuals           #
# ATENTION:                                                         #
#    1- Vandergast et al. (2008) suggests a RMA instead of a        #
#       ordinary linear regression as in this example. Try package  #
#       'lmodel2' or or other similar for RMA linear regression.    #
#    2- This example tests if the package 'geometry' is installed   #
#       to compute midpoints. If TRUE, a Delaunay triangulation is  #
#       used, similarly to Vandergast et al. (2008). Otherwise,     #
#       midpoints are computed for the combination of all pairs of  #
#       samples.                                                    #
#                                                                   #
# the d.gen and d.real matrices in this example have the same       #
# column and row order!                                             #
#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#

if (is.element('geometry', installed.packages()[,1]))
    all=FALSE else
```

```
    all=TRUE

mp <- midpoints(vipers[,1:2], all=all)
d.real <- as.matrix(dist(vipers[,1:2]))

fit <- lm(as.vector(d.gen) ~ as.vector(d.real))
resid <- matrix(fit$residuals, nrow(vipers), nrow(vipers))
dimnames(resid) <- dimnames(d.gen)
mp$z <- extract.val(resid, mp[,1:2])

int <- idw(mp[,5], mp[,3:4], grid)

grid.image(int, grid, main='IDW interpolation',
           xlab='Longitude', ylab='Latitude',
           sclab="Residuals of genetic vs. real distances")

# plot samples connecting lines
for (i in 1:nrow(mp))
{
    pair <- as.character(unlist(mp[i,1:2]))
    x <- c(vipers[pair[1],1], vipers[pair[2],1])
    y <- c(vipers[pair[1],2], vipers[pair[2],2])
    lines(x, y, lty=2)
}
points(vipers[,1:2], pch=16) # plot samples points in black
points(mp[,3:4], pch=16, col='gray') # plot midpoints in gray
```

`intgen.idw` *Interpolation of genetic distances to a a grid of points.*

Description

Interpolations of a matrix containing genetic distances using the Inverse Distance Weighting (IDW) algorithm. It generates a matrix of interpolated values for each grid cell and for each sample.

Usage

```
intgen.idw(d.real, d.gen, method = "Shepard", p = 2, R = 2, N = 15)
```

Arguments

`d.real`	distance matrix between sampled locals (columns) and locals where interpolation is to be executed (rows). Names should correspond to genetic distances matrix.
`d.gen`	genetic distances matrix. Names should correspond to real distances matrix, but not necessarily in the same order.

method	Method to calculate weights for idw. Should be "Shepard" (default), "Modified", "Neighbours", or distinctive abreviations of each. See details section for additional help on each method.
p	The power to use in weight calculation.
R	Radius to use with Modified Shepard method.
N	Maximum number of neighbours to use with Shepard with neighbours.

Details

The IDW interpolation algorithm is commonly used to interpolate genetic data over a spatial grid. This function provides a simple interface to interpolate such data with three methods:

1. *Shepard*: weights are the inverse of the distance between the interpolation location x and the sample points x_i, raised to the power p

$$w(x) = \frac{1}{d(x, x_i)^p}$$

2. *Modified Shepard*: distances are weighted with a search radius r to calculate the interpolation weights

$$w(x) = \left(\frac{r - d(x, x_i)}{r.d(x, xi)} \right)^p$$

3. *Shepard with neighbours*: A maximum ammount of N neighbours is allowed to the weight calculation following Shepard method.

The functions 'intgen.idw' and 'idw' are similar but whereas the first interpolate all samples to a grid-based coordinates, the second interpolates a single set of values. Although 'idw' can be used to generate the same results, the 'intgen.idw' should be faster (see the examples).

Value

This function returns a matrix containing all interpolated values for each locality (rows) and for each sample (columns)

Author(s)

Pedro Tarroso <ptarroso@cibio.up.pt>

References

Fortin, M. -J. and Dale, M. (2006) *Spatial Analysis: A guide for Ecologists*. Cambridge: Cambridge University Press.

Isaaks, E. H. and Srivastava, R. M. (1989) *An Introduction to applied geostatistics*. New York: Oxford University Press.

Legendre, P. and Legendre, L. (1998) *Numerical ecology*. 2nd english edition. Amesterdam: Elsevier

See Also

idw

Examples

```
data(vipers)
data(d.gen)
data(grid)

# create a matrix of distances from sample points (columns) to all
# grid pixels
rd <- geo.dist(grid, vipers[,1:2])

#interpolate with idw
result <- intgen.idw(rd, d.gen)

#plot the 12 random interpolations
layout(matrix(c(1:12), 4,3))

for (i in sample(1:nrow(vipers), 12))
{
    dt <- data.frame(grid, int=result[,i])
    # when samples are given with real coordinates, aspect of image
    # should be maintained with asp=1 to plot properly.
    image(sort(unique(grid[,1])), sort(unique(grid[,2])),
          xtabs(int~x+y, dt), xlab='Longitude', ylab='Latitude',
          main=colnames(result)[i])
    cex <- (d.gen[,i]-min(d.gen[,i]))/(max(d.gen[,i])-min(d.gen[,i]))
    points(vipers[,1:2], cex=cex+0.5)
}

## Not run:
    # Compare 'idw' with 'intgen.idw' to generate the same results.
    # NOTE: it may take a few minutes to run.
    result2 <- matrix(NA, nrow=nrow(grid), ncol=nrow(vipers))
    for (i in 1:nrow(vipers)) {
        values <- d.gen[i,]
        intpl <- idw(values, vipers[,1:2], grid)
        result2[,i] <- intpl[,1]
    }
    colnames(result2) <- rownames(vipers)

    # compare all items in the two matrices to check equality:
    all(result == result2)

## End(Not run)
```

---

krig *Simple and ordinary kriging.*

Description

Computes simple or ordinary kringing using a list of sampled locations. The interpolation is executed to the table of coordinates given.

Usage

```
krig(values, coords, grid, gv, distFUN = geo.dist, ..., m=NA, cv=FALSE,
     neg.weights = TRUE, clamp = FALSE, verbose = TRUE)
```

Arguments

values	A vector of values per sampled location.
coords	A table containing longitude and latitude of sample locations for each value.
grid	A table containing the coordinates of locations to interpolate
gv	A fitted model to variogram as given by 'gv.model' function.
distFUN	Distance function used to calculate distances between locations. The default is 'geo.dist' which calculates simple euclidean distances between the locations. This function must have a 'from' and a 'to' arguments to specify, respectively, the source and destination localities. Other functions can be given to include, for instance, a landscape resistance (see examples and vignette).
...	Other arguments to be passed to distFUN.
m	A value for the mean. When the mean is known and given, a simple kriging is used for the interpolation. If m = NA (default) then the mean is estimated using ordinary kriging.
cv	A logical value to perform cross validation of the interpolation.
neg.weights	A logical value indicating if negative weights are allowed in the calculation. If FALSE, negative weights are corrected and eliminating the need for clamping. See details.
clamp	A logical value indicating if Z values will be adjusted to the interval [0,1].
verbose	A logical indicating if the function should be verbose.

Details

This function interpolates the probability of lineage occurrence to all locations given in 'coords'. Usually 'coords' stores coordinates of pixel centroids representing the study area with a user-defined spatial resolution. The variogram with a fitted model describes the autocorrelation structure of the genetic data. This is used by kriging to determine the weight of the sampled points on the location to predict a value.

The most usual kriging is performed with geographical distances between localities (samples and grid). The default function is the 'geo.dist' that calculate simple euclidean distances. However, the 'krig' function allows to provide user-defined distance functions to calculate other kind of distances. A landscape resistance, for instance, may be used to calculate cost distances between points and to be used in the interpolation process. The same distance function provided here has to be used before, to produce the variogram. The function 'distFUN' has to have the arguments 'from' and 'to' to use as source and destination coordinates, respectively. It may have other arguments that can be passed to the function. See the vignette for a exhaustive example on how to use this functionality.

The kriging algorithm often produce negative weights for the interpolation. This results in predictions outside the original range between zero and one. Correcting negative weights allows to maintain predictions in this range and to preserve kriging proprieties. Correction of negative weights is

done following Deutsch (1996): negative weights and small positive weights (within the variation of the negative weights) are set to zero and the sum of the resulting weights rescaled to one.

Cross-validation is computed by leaving each of the observation in 'values' out of kriging and predict for the same location. A mean squared error (MSE) is computed using the original observation and the predicted value.

Value

Returns a vector of interpolated values and respective variance for each location in 'coords'.

If cross-validation is performed (cv=TRUE) than a list of interpolation and variance values is given with a cross-validation matrix (original observation and predicted value) and a mean squared error (MSE).

Author(s)

Pedro Tarroso <ptarroso@cibio.up.pt>

References

Deutsch C. V. (1996) *Correcting for negative weights in ordinary kriging*. Computers and Geosciences, 22(7), 765-773.

Fortin, M. -J. and Dale, M. (2006) *Spatial Analysis: A guide for Ecologists*. Cambridge: Cambridge University Press.

Isaaks, E. H. and Srivastava, R. M. (1989) *An Introduction to applied geostatistics*. New York: Oxford University Press.

Legendre, P. and Legendre, L. (1998) *Numerical ecology*. 2nd english edition. Amesterdam: Elsevier

See Also

`gen.variogram` `plot.gv` `predict.gv` `idw` `intgen.idw`

Examples

```
data(vipers)
data(d.gen)
data(grid)

# In this example we want to create the probable distribution of a
# lineage based on the genetic distance. We need a vector defining if
# each sample belongs or not to the lineage
lin <- as.integer(vipers$lin == 1)

# create a distance matrix between samples
r.dist <- dist(vipers[,1:2])

# fit a model with defaults (spherical model) and estimation of range
gv <- gen.variogram(r.dist, d.gen)
gv <- gv.model(gv)
```

```
# perform interpolation with ordinary kriging
int.krig <- krig(lin, vipers[,1:2], grid, gv)

#plot the interpolation results
grid.image(int.krig, grid, main='Kriging with genetic distances',
           xlab='Longitude', ylab='Latitude',
           sclab='Lineage interpolation')
points(vipers[,1:2], pch=lin+1)

# plot the interpolation standard variance
grid.image(int.krig, grid, ic='sd',
           main='Kriging with genetic distances',
           xlab='Longitude', ylab='Latitude',
           sclab='Standard deviation')
points(vipers[,1:2], pch=lin+1)

#plot only pixels higher than 0.95
lin.krig <- as.integer(int.krig$Z>0.95)
grid.image(lin.krig, grid, main='Kriging with genetic distances',
           xlab='Longitude', ylab='Latitude', sclab='Lineage')
points(vipers[,1:2], pch=lin+1)

## Not run:
#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#
# The following code is an example to combine the possible clusters   #
#  of a tree in a single map. It samples the tree at different length #
# and combines the probabilities.                                     #
#                                                                     #
#                NOTE: it may take some time to run!                  #
#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#

# A phylogenetic tree is sampled at different lengths. An appropriate
# package (e.g. ape) should be used to process the tree. To avoid extra
# dependencies we convert the genetic distances to an hclust.

hc <- hclust(as.dist(d.gen))
hs = seq(0.01, 0.08, 0.005) # tree length sampling

# Another options for tree sampling can be using the nodes position,
# avoiding the root and tip levels:
# hs <- hc$height[hc$height > 0.0 & hc$height < max(hc$height)]
#
# Or a single threshold:
# hs <- 0.06

contact = rep(0, nrow(grid)) # Sums all probabilities
for (h in hs) {
    lins <- cutree(hc, h=h)
    print(paste("height =", h, ":", max(lins), "lineages")) #keep track
    ct = rep(1, nrow(grid)) # Product of individual cluster/lineage map
    for (i in unique(lins)) {
        lin <- as.integer(lins == i)
```

```
        krg <- krig(lin, vipers[,1:2], grid, gv,
                    clamp = TRUE, verbose=FALSE)
        # Probability of NOT belonging to a cluster.
        ct <- ct * (1 - krg$Z) # Probab. of NOT belonging to a cluster
    }
    contact = contact + ct
}
krg$Z <- contact / length(hs) # Recycle krg with contact zones

#plot the interpolation results
grid.image(krg, grid, xlab='Longitude', ylab='Latitude',
           main='Uncertainty in cluster classification / contact zones')
points(vipers[,1:2], pch=16, cex=0.5)

## End(Not run)
```

`midpoints` *Midpoints between pairs of coordinates*

Description

Computes the midpoints for a table of sample points with coordinates.

Usage

```
midpoints(samples, x=1, y=2, sp.name=row.names(samples), all=FALSE)
```

Arguments

`samples`	Table with coordinates for each sample point.
`x`	Column index or name of longitudes (x) in samples table (default is first column).
`y`	Column index or name of latitudes (y) in samples table (default is second column).
`sp.name`	Name for each sample point (defaults to row names of samples).
`all`	If TRUE computes midpoints between all sample points. If FALSE (default) computes a Delaunay triangulation and the midpoints of the resulting connected samples.

Details

This function computes the coordinates of the middle points between samples. The connecting network can be between all points or between neighbours with non-overlaping edges after a Delaunay triangulation.

Value

Returns a data frame with 4 columns referring the source and target samples (ss and ts, respectively) and the coordinates of the midpoints.

Note

Depends on package 'geometry' for Delaunay triangulation.

Author(s)

Pedro Tarroso <ptarroso@cibio.up.pt>

See Also

dist d.gen extract.val

Examples

```
data(vipers)
mp <- midpoints(vipers[,1:2], all=TRUE)
# With 'all=FALSE' (Delaunay triang.), package 'geometry' is mandatory.
```

---

mpinv *Generalized inverse of a matrix*

Description

Computes the generalized inverse of a matrix using singular-value decomposition.

Usage

```
mpinv(A, eps = 1e-13)
```

Arguments

A	Matrix to be inverted.
eps	Minimum value threshold.

Value

Returns a matrix containing the inverse of matrix A.

Author(s)

Pedro Tarroso <ptarroso@cibio.up.pt>

Examples

```
m <- matrix(rnorm(16), 4, 4)
mi <- mpinv(m)
```

mtest.gv *Tests if a 'gv' object has a model.*

Description

Tests if the semi-variogram in the 'gv' object has a model fitted.

Usage

```
mtest.gv(gv)
```

Arguments

gv	An object with class 'gv'.

Details

This function is usefull to test if a model for the semi-variogram in 'gv' is already built.

Value

logical

Note

It is not exported.

Author(s)

Pedro Tarroso <ptarroso@cibio.up.pt>

multispecies *Summarizes data from multiple species.*

Description

This function may be used to summarize data from different species or similar (e.g. different lineages, etc). By default, it summarizes with the mean and standard deviation, but different functions may be used.

Usage

```
multispecies(..., FUN=list(mean=mean, sd=sd), na.rm=FALSE)
```

Arguments

...	Input data to be summarized. It must be numeric and it can be multiple vectors of the same size and order, or more simply a matrix with all data to be summarised. The matrix must have the different species/data in columns.
FUN	This is a list of functions to be applied to summarize the data. By default it uses the mean and sd, but it can be any other function that returns a number from a vector (e.g. max, min) or a user-defined function. If the objects are named in the FUN list, than those names will be given to the resulting columns. Otherwise, function are applied in the same order as given.
na.rm	A logical indicating whether missing values should be removed. Will only work if the functions in FUN accept it.

Details

This function is a simple wrapper with some error checking for the native R function 'apply'.

Value

Returns a matrix with functions applied in the same order as FUN.

Author(s)

Pedro Tarroso <ptarroso@cibio.up.pt>

See Also

apply princomp prcomp

Examples

```
data(vipers)
data(d.gen)
data(grid)

# create a matrix of distances from sample points (columns) to all
# grid pixels
rd <- geo.dist(grid, vipers[,1:2])

#interpolate with idw
result <- intgen.idw(rd, d.gen)

ms <- multispecies(result)

# plot the mean
grid.image(ms, grid, main = "Mean")

# plot the standard deviation
grid.image(ms, grid, ic=2, main = "Standard Deviation")
```

plot.gv *Plot a 'gv' object*

Description

Plot the semi-variogram in a gv object. If a multiple genetic distances are found, it plots the median value and the 95% confidence interval for the median.

Usage

```
## S3 method for class 'gv'
plot(x, line.res = 100, pch=1, legend=TRUE, leg.x=NA,
                  leg.y=NA, leg.cex=1, bar.length=0.1, bar.col="gray",
                  bar.lty=par("lty"), xlab='Distance', ylab='Semivariance',
                  x.line=3, y.line=3, ncol=1, main=NULL,
                  leg.label = expression(italic('n')*' size'), ...)
```

Arguments

x	'gv' object as given by 'gen.variogram'.
line.res	Number of points in the model line.
pch	Symbol to be used in the plot.
legend	Boolean indicating if a legend showing *n* size should be printed.
leg.x	The x position for the legend. The legend will be placed at the right side of the plot if this value is set to NA.
leg.y	The y position for the legned. The legend will be placed at the bottom of the plot if this value is set to NA.
leg.cex	Multiplication factor for the legend symbol size.
bar.length	If multiple trees are given, confidence interval bars are ploted. The horizontal length of the line at both bar tips is defined with this parameter (defaults to 0.1).
bar.col	The color of the bars when multiple trees are given.
bar.lty	The line type for the bars when multiple tree are given.
xlab	The label for x axis.
ylab	The label for y axis.
x.line	Position of x label in lines.
y.line	Position of y label in lines.
ncol	Number of legend columns.
main	Main title of the plot.
leg.label	Legend title.
...	Further plotting arguments to be passed.

Details

Simple plot of the semi-variogram contained in a 'gv' object. If the object has a model, the model line is also plotted.

Value

Plot.

Author(s)

Pedro Tarroso <ptarroso@cibio.up.pt>

See Also

gen.variogram

Examples

```
data(vipers)
data(d.gen)

# create a distance matrix between samples
r.dist <- dist(vipers[,1:2])

# fit a variogram with defaults (shperical model) and estimation of range
gv <- gen.variogram(r.dist, d.gen, 0.25)

#plot semi-variogram
plot(gv)

# plot semi-variogram with model
gv <- gv.model(gv)
plot(gv)
```

---

predict.gv *Predict method for 'gen.variogram' object with model.*

Description

Predicts values based on a fitted gen.variogram model.

Usage

```
## S3 method for class 'gv'
predict(object, newdata, ...)
```

Arguments

object	'gv' fitted model (see 'gen.variogram').
newdata	Real distances matrix to predict genetic distance by the fitted model.
...	Further arguments to be passed.

Value

Returns the matrix of predicted genetic distances.

Author(s)

Pedro Tarroso <ptarroso@cibio.up.pt>

See Also

gen.variogram plot.gv krig

Examples

```
data(vipers)
data(d.gen)

# create a grid of the sampled area for interpolation
grid <- expand.grid(x=seq(-10,10,0.5), y=seq(30, 50, 0.5))

# create a distance matrix between samples
r.dist <- dist(vipers[,1:2])

# fit a variogram with defaults (spherical model) and estimation of range
gv <- gen.variogram(r.dist, d.gen)
gv <- gv.model(gv)

all.dist <- as.matrix(dist(grid))

result <- predict(gv, all.dist)
```

---

print.gv *Prints details of a 'gv' object*

Description

The function is used when a 'gv' object is called.

Usage

```
## S3 method for class 'gv'
print(x, ...)
```

Arguments

x	'gv' object as given by 'gen.variogram'.
...	Further plotting arguments to be passed.

Details

This prints the details of a 'gv' object including number of observations and other variogram creation parameters used. It will also display model details if a model was fitted to the empirical variogram.

Author(s)

Pedro Tarroso <ptarroso@cibio.up.pt>

See Also

gen.variogram

Examples

```
data(vipers)
data(d.gen)

# create a distance matrix between samples
r.dist <- dist(vipers[,1:2])

# fit a variogram with defaults (shperical model) and estimation of range
gv <- gen.variogram(r.dist, d.gen, 0.25)

# print variogram details
gv

# add a model to variogram
gv <- gv.model(gv)

# print variogram with model details
gv
```

---

simul.env *Simulated environments.*

Description

A table with two gridded simulation environments.

Usage

```
data(simulations)
```

Format

'simul.env' is a data frame with 1849 rows and 4 columns. Each row is a cell in the gridded surface. The column one and two are the x and y centroid coordinates of each cell, respectively. The third and fourth columns are the two simulates environments.

References

Tarroso, P., Carvalho, S.B. & Velo-Anton, G. (2019). PHYLIN 2.0: Extending the phylogeographic interpolation method to include uncertainty and user-defined distance metrics. Molecular Ecology Resources, in press.

Examples

```
data(simulations)
# Plot the second environmental surface: negative values as open circles and
# positive as solid black circles. Size proportional to absolute value
plot(simul.env[,1:2], pch=1, cex=simul.env[,4]/5)
points(simul.env[,1:2], pch=16, cex=-simul.env[,4]/5)
```

---

simul.gen.dist *Simulated genetic distances.*

Description

A 'dist' class object with simulated genetic distances.

Usage

```
data(simulations)
```

Format

'simul.gen.dist' is an object with 'dist' class containing the simulated genetic distances of 200 samples.

References

Tarroso, P., Carvalho, S.B. & Velo-Anton, G. (2019). PHYLIN 2.0: Extending the phylogeographic interpolation method to include uncertainty and user-defined distance metrics. Molecular Ecology Resources, in press.

Examples

```
data(simulations)
hc <- hclust(as.dist(simul.gen.dist))
plot(hc, main="Simulated gentic distances", xlab="Samples", cex=0.7)
```

simul.sample *Random samples from simulation.*

Description

A table with x and y coordinates of 200 random samples in the simulated grid and a lineage information.

Usage

```
data(simulations)
```

Format

'simul.sample' is a data frame with 200 rows and 3 columns. Each row is a sample in the simulated grid. The column one and two are the x and y coordinates of each sample, respectively. The lineage to which each sample belongs is given in the third column.

References

Tarroso, P., Carvalho, S.B. & Velo-Anton, G. (2019). PHYLIN 2.0: Extending the phylogeographic interpolation method to include uncertainty and user-defined distance metrics. Molecular Ecology Resources, in press.

Examples

```
data(simulations)
# Plot all the samples with different symbols for each lineage.
plot(simul.sample[,1:2], pch=simul.sample[,3])
```

summary.gv *Summary for 'gv' object*

Description

Displays general information about the 'gv' object.

Usage

```
## S3 method for class 'gv'
summary(object, ...)
```

Arguments

object	'gv' object as given by 'gen.variogram' or 'gv.model'.
...	Further plotting arguments to be passed.

Value

Print summary table.

Author(s)

Pedro Tarroso <ptarroso@cibio.up.pt>

See Also

gen.variogram gv.model

Examples

```
data(vipers)
data(d.gen)

# create a distance matrix between samples
r.dist <- dist(vipers[,1:2])

# fit a variogram with defaults (shperical model) and estimation of range
gv <- gen.variogram(r.dist, d.gen)

#plot semi-variogram
summary(gv)

# plot semi-variogram with model
gv <- gv.model(gv)
summary(gv)
```

---

vipers *Vipers sample locations for 'd.gen' dataset.*

Description

This dataset contains the x and y coordinates of 58 *Vipera latastei* samples with corresponding lineages.

Usage

```
data(vipers)
```

Format

A data frame with 3 columns (x/Longitude, y/Latitude and lineage) and 58 rows.

Source

Velo-Anton G., Godinho R., Harris D. J. *et al.* (2012) Deep evolutionary lineages in a Western Mediterranean snake (*Vipera latastei/monticola* group) and high genetic structuring in Southern Iberian populations. *Molecular phylogenetics and evolution*, **65**, 965–973.

Examples

```
data(vipers)
data(grid)
plot(grid, cex=0.5, col='lightgrey', asp=1,
     main="Vipers data", xlab="Longitude", ylab="Latitude")
points(vipers[,1:2], pch=vipers$lin)
legend(1, 38, legend=c("West", "South", "East"), pch=1:3, title="Lineages")
```

Index