Package ‘ggene’

June 7, 2016

Type Package

Title Semivariogram and Exploratory Spatial Analysis of Genetic Data

Version 1.0.2

Date 2016-05-31

Author Jean-Pierre Rossi

Maintainer Jean-Pierre Rossi <`ggene.package@gmail.com`>

Depends R (>= 3.1.0)

Imports geoR, FactoMineR, adegenet (>= 2.0.0), spatstat, sp, gstat, plotrix

Description Semivariogram and spatial analysis of genetic data: semi-variance computation, model fitting, analysis of anisotropy.

License GPL (>= 2)

LazyData true

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

R topics documented:

---

ggene-package *Semivariogram and exploratory spatial analysis of genetic data*

Description

Semivariogram and spatial analysis of genetic data: semivariance computation, model fitting, analysis of anisotropy. A good place to start is the vignettes, which you can access by typing `vignette("ggene_introduction")` and `vignette("ggene_datasets")`.

Details

Package:	ggene
Type:	Package
Version:	1.0.2
Date:	2016-05-31
License:	GPL (>= 2)

ggene was designed to provide a variety of tools allowing to analyse microsatellite data recorded for geolocated individuals. The package largely relies on geostatistics. Amongst the numerous textbooks dedicated to geostatistical analysis, readers are referrred to Goovaerts (1997) and Isaaks and Srivastava (1989). Diggle and Ribeiro (2007) offer a very good introduction to geostatitsics in the `R` environment. In their seminal work, Wagner et al (2005) introduced the use of semivariograms to analyse the spatial variation of genetic data and ggene implements the corresponding methods and introduce new tools such as variogram maps.

Author(s)

Jean-Pierre Rossi <ggene.package@gmail.com>

References

Diggle, P., P. J. Ribeiro. 2007. Model-Based Geostatistics. Springer.

Goovaerts, P. 1997. Geostatistics for Natural Resources Evaluation. Oxford University press.

Isaaks, E. H., R. M. Srivastava. 1989. Applied Geostatistics. Oxford University Press.

Wagner, H. H., R. Holderegger, S. Werth, F. Gugerli, S. E. Hoebee and C. Scheidegger. 2005. Variogram analysis of the spatial genetic structure of continuous populations using multilocus microsatellite data. Genetics 169, 1739-1752.

Examples

```
### variogram computation #########################
# omnidirectional variogram: simple computation
data(larix2300)
```

```
va <- svariog(X=larix2300, plot=FALSE)

# plotting the variogram
plot(va$svario$u, va$svario$v)

# another example
data(larix1350)
va3 <- svariog(X=larix1350, uvec=distlag(dist=larix1350$coord, dmin=0, distance.lag=10),
plot=FALSE)
plot(va3$svario$u, va3$svario$v)

### statistical envelopes #########################
## Not run:
# computing and plotting statistical envelopes
env <- randsvariog(var=va3, X=larix1350, nsim=30,
  bounds=c(0.025, 0.975), save.sim=FALSE)
plot(env$svario$u, env$svario$v, ylim=range(env$env),
     xlab="distance", ylab="semi-variance")
points(env$svario$u, env$env[,1], type="l")
points(env$svario$u, env$env[,2], type="l")

## End(Not run)

### computation of the directional variogram #########################
## Not run:
data(aniso)
va <- svariog(X=aniso, plot=TRUE)

d0_225 <- svariog(X=aniso,direction=0, tolerance=22.5, unit.angle="degrees")
d45_225 <- svariog(X=aniso,direction=45, tolerance=22.5,  unit.angle="degrees")
d90_225 <- svariog(X=aniso,direction=90, tolerance=22.5,  unit.angle="degrees")
d135_225 <- svariog(X=aniso,direction=135, tolerance=22.5,  unit.angle="degrees")

plot(va$svario$u, va$svario$v, type="b", ylim=range(c(va$svario$v,d0_225$svario$v,
d45_225$svario$v, d90_225$svario$v, d135_225$svario$v)), xlab="distance",
ylab="semivariance")

points(d0_225$svario$u, d0_225$svario$v, type="b", lty=2)
points(d45_225$svario$u, d45_225$svario$v, type="b", col="red", lty=2)
points(d90_225$svario$u, d90_225$svario$v, type="b", col="blue", lty=2)
points(d135_225$svario$u, d135_225$svario$v, type="b", col="green", lty=2)

legend("topleft", legend=c("omnidirectional", expression(0 * degree), expression(45 * degree),
  expression(90 * degree), expression(135 * degree)), lty=c(1,2,2,2,2,2),
col=c("black", "black","red","blue","green"), bty="n")

## End(Not run)


# variogram maps
data(aniso)
map <- svarmap(X=aniso,cutoff=20, width=1)
plot(map)


### omnidirectional variogram: weighting for recurrent genotypes #########################
data(crypho)
```

```
# compute matrix of weights
count <- genocount(X=crypho)
mat <- genoweight(X=crypho,genotyp=count$vec)
d <- distlag(dist=crypho$coord, dmin=0,distance.lag=50)

# compute variogram
wva <- varioWeight(X=crypho, weights=mat,  uvec=d)

#plot the weighted variogram
plot(wva$svario$u, wva$svario$gamma, col="black", type="b",
     ylim=range(c(wva$svario$gamma,wva$svario$v)))

#add the variogram for raw data
points(wva$svario$u, wva$svario$v, col="red", type="b")

### fitting models #########################
data(sim03)
va <- svariog(X=sim03, plot=TRUE)
fit <- fitsvariog(vario=va, ini.cov.pars=c(0.05,4.5), nugget=0.5, max.dist=200)
fit

# graphical display
fit <- fitsvariog(vario=va, ini.cov.pars=c(0.05,4.5), nugget=0.5, max.dist=200, plot=FALSE)
plot(va$svario$u, va$svario$v)
lines(fit$fit)
```

---

aniso *A simulated haploid genotypic data set exhibiting zonal anistropy*

Description

aniso provides a simulated dataset illustrating how variogram can be used to explore spatial anisotropies in genetic data. Functions svariog and svarmap can be used to compute the variogram in different directions and determine if it behaves differently in some of them. If the variogram changes according to directions, it is said to be anisotropic. If not, the variogram is said to be isotropic. The dataset aniso is an object of class ggene with 400 individuals described by means of 3 locus for a total of 13 alleles.

Usage

```
data(aniso)
```

Format

An object of class ggene.

Examples

```
data(aniso)

# checking anisotropy using a variogram map
```

```
map <- svarmap(X=aniso,cutoff=20, width=1) ; plot(map)

# compute omnidirectional and directional variograms
va <- svariog(X=aniso, plot=TRUE)

d0_225 <- svariog(X=aniso,direction=0, tolerance=22.5, unit.angle="degrees")
d45_225 <- svariog(X=aniso,direction=45, tolerance=22.5, unit.angle="degrees")
d90_225 <- svariog(X=aniso,direction=90, tolerance=22.5, unit.angle="degrees")
d135_225 <- svariog(X=aniso,direction=135, tolerance=22.5, unit.angle="degrees")

plot(va$svario$u, va$svario$v, type="b", ylim=range(c(va$svario$v, d0_225$svario$v,
  d45_225$svario$v, d90_225$svario$v, d135_225$svario$v)) ,xlab="distance",
  ylab="semi-variance")

points(d0_225$svario$u, d0_225$svario$v, type="b", lty=2)
points(d45_225$svario$u, d45_225$svario$v, type="b", col="red", lty=2)
points(d90_225$svario$u, d90_225$svario$v, type="b", col="blue", lty=2)
points(d135_225$svario$u, d135_225$svario$v, type="b", col="green", lty=2)

legend("topleft", legend=c("omnidirectional", expression(0 * degree),
    expression(45 * degree), expression(90 * degree),
    expression(135 * degree)), lty=c(1,2,2,2,2,2),
    col=c("black","black","red","blue","green"), bty="n")
```

`crypho` — *A haploid microsatellite dataset for the chestnut blight fungus* Cryphonectria parasitica

Description

A set of 10 locus for 276 individuals of the chestnut blight fungus *Cryphonectria parasitica*.

Usage

```
data(crypho)
```

Format

An object of class ggene.

Note

The coordinates of the individuals were slightly jittered (a few centimeters) because some individuals were superimposed. Jittering removed the duplicated points, hence the various warning messages issued by `svariog`. There is no consequences on the variograms because jeterring implied distances much lower than lag distance.

Source

Dutech, C., J.-P. Rossi, O. Fabreguettes and C. Robin 2008. Geostatistical genetic analysis for inferring the dispersal pattern of a partially clonal species: example of the chestnut blight fungus. Molecular ecology 17: 4597-4607.

Examples

```
data(crypho)

# check sampling scheme
plot(crypho$coord[,1],crypho$coord[,2], asp=1)

# compute matrix of weights
count <- genocount(X=crypho)
mat <- genoweight(X=crypho,genotyp=count$vec)

# compute distance intervals
d <- distlag(dist=crypho$coord, dmin=0,distance.lag=50)

# compute weighted variogram
wva <- varioWeight(X=crypho, weights=mat,  uvec=d)

# plot the variogram for raw data
plot(wva$svario$u, wva$svario$gamma, col="black", type="b",
ylim=range(c(wva$svario$gamma,wva$svario$v)), xlab="distance", ylab="semivariance")


# add the weighted variogram
points(wva$svario$u, wva$svario$v, col="red", type="b", pch=4)

legend("top", legend=c("raw", "weighted"), col=c("black", "red"), lty="solid", pch=c(1,4), bty="n")


## Not run:

#performs randomization on raw variogram
va <- svariog(X=crypho, plot=FALSE)
env <- randsvariog(var=va, X=crypho, nsim=9, bounds=NULL, save.sim=FALSE)

#compute the weighted variogram
wva <- varioWeight(X=crypho, weights=mat)

#performs the randomizations on weighted variogram
env2 <- randsvariog(var=wva, X=crypho, nsim=9, bounds=NULL, save.sim=FALSE, weights=mat)

# plot results
xx <- c(wva$svario$u, rev(wva$svario$u))
yy <- c(env$env[,1], rev(env$env[,2]))
plot(xx, yy, type = "n", xlab = "distance", ylab = "semivariance",
 ylim=range(c(env$env[,1], env$env[,2], env2$env[,1], env2$env[,2])))
polygon(xx, yy, col = "lightgrey", border = "black")
xx <- c(wva$svario$u, rev(wva$svario$u))
yy <- c(env2$env[,1], env2$env[,2])
points(xx, yy, type = "l")
polygon(xx, yy, col = "lightblue", border = "blue")

points(wva$svario$u, wva$svario$v, col="blue", typ="b")
points(wva$svario$u, wva$svario$gamma, col="black", type="b", lty="solid", bty="n")


## End(Not run)
```

```
# fit exponential model to empirical variogram
va <- svariog(X=crypho, plot=TRUE, messages=FALSE)
fit <- fitsvariog(vario=va, ini.cov.pars=c(0.03,100), nugget=0.1, max.dist=300, plot = TRUE)
fit$param

###

# compute variogram map
 map <- svarmap(X=crypho,cutoff=1000, width=50) ; plot(map)
```

---

distlag *Compute custom distance lags*

Description

The function computes the centre of a set of distance classes from a (geographical) distance matrix or a data frame containing the point coordinates. The minimum, maximum and interval distances can be customized. The function returns a vector that can be used to feed the function `svariog`.

Usage

```
distlag(dist, dmin = 0, distance.lag = NULL, dist.lag.max = NULL)
```

Arguments

`dist`	A `data.frame` with 2 columns containing the point coordinates or a distance matrix (class `dist`).
`dmin`	The minimum distance to be considered. Default is 0.
`distance.lag`	The distance increment between two successive centre of distance classes. A default value is computed on the basis of the function `hist`.
`dist.lag.max`	The maximum distance to be considered.

Details

`distlag` creates a set of bins describing the distance classes on the basis of the point spatial location and a user-defined lag interval. Spatial information is provided as a distance matrix (class `dist`) or a set of coordinates (`data.frame`). Data frame must contain 1 or 2 columns (i.e. data points in 1 or 2 dimensions). NAs are not allowed and should be removed prior to using the function.

Value

A vector of values corresponding to the centre of successive distance classes

Author(s)

Jean-Pierre Rossi <ggene.package@gmail.com>

Examples

```
data(sim03)

# check sampling scheme
plot(sim03$coord[,1],sim03$coord[,2], asp=1)

# changing the distance interval
distlag(dist=sim03$coord,dmin=0, distance.lag=0.5, dist.lag.max=NULL)
distlag(dist=sim03$coord,dmin=0, distance.lag=2, dist.lag.max=NULL)

# changing the maximum distance to be considered
x <- y <- seq(0,10, length.out=10)
coord <- expand.grid(x=x, y=y)
distlag(dist=coord,dmin=0.5, distance.lag=1, dist.lag.max=NULL)
distlag(dist=coord,dmin=0.5, distance.lag=1, dist.lag.max=10)

# using a distance matrix
m<-dist(coord)
distlag(dist=m)
```

`fitsvariog` *Fit an exponential model to empirical variogram*

Description

The function fits an exponential model to empirical variogram and provides the estimates of the covariance parameters.

Usage

```
fitsvariog(vario, ini.cov.pars, plot = TRUE, ...)
```

Arguments

`vario`	an object of class `svariog`, typically an output of the function `svariog`.
`ini.cov.pars`	initial values for variogram model: σ^2 (partial sill) and ϕ (range parameter). See `variofit` from package `geoR`.
`plot`	logical, if TRUE, the empirical variogram is plotted with the fitted model and the value of the conventional estimate of the genetic diversity (Hhat in wagner et al 2005).
`...`	additional parameters passed to `variofit` (package `geoR`). SEE DETAILS SECTION BELOW.

Details

The function calls the function `variofit` from package `geoR` (Diggle and Ribeiro, 2007). It fits an exponential model to the empirical variogram and returns the estimated parameters. In addition, the parameters FN, bf and Sp (Vekemans and Hardy, 2004) are estimated from the fitted variogram following Wagner et al. (2005). The optional arguments are :

cov.model variogram model to be fitted. The only available model the the exponential model.

fix.nugget logical, indicating whether the parameter τ^2 (nugget variance) should be considered as fixed (`fix.nugget = TRUE`) or should be estimated (`fix.nugget = FALSE`). See `variofit` from package `geoR`.

nugget value for the nugget parameter. Defaults set to zero. See `variofit` from package `geoR`.

max.dist maximum distance considered when fitting the variogram. Defaults set to `svario$max.dist`. See `variofit` from package `geoR`.

Value

`param` a vector giving the fitted parameters

- c: spatial variance aka partial sill
- nugget: nugget variance
- range: range
- pract.range: practical range
- sill: sill variance
- Hhat: conventional estimate of gene diversity (non spatial estimate analogous to the variance)
- FN: Relatedness of immediate neighbors (see Vekemans and Hardy (2004)
- bf: A slope parameter involved is the estimation of `Sp`
- Sp: The index of spatial Genetic Structure proposed by Vekemans and Hardy (2004)

`fit` an object of class `variofit` from the package `geoR`

Author(s)

Jean-Pierre Rossi Jean-Pierre Rossi <ggene.package@gmail.com>

References

Diggle, P. J. and P. J. Ribeiro 2007. Model-based Geostatistics, Springer.

Vekemans, X. and O. J. Hardy. 2004. New insights from fine-scale spatial genetic structure analyses in plant populations. Molecular Ecology 13: 921-935.

Wagner, H. H., R. Holderegger, S. Werth, F. Gugerli, S. E. Hoebee and C. Scheidegger. 2005. Variogram analysis of the spatial genetic structure of continuous populations using multilocus microsatellite data. Genetics 169, 1739-1752.

See Also

`svariog`

Examples

```
## fit model to empirical variograms from simulated datsets
data(aniso)
va <- svariog(X=aniso, plot=FALSE)
fit <- fitsvariog(vario=va, ini.cov.pars=c(0.05,10), nugget=0.2, max.dist=30, plot = TRUE)
fit

data(sim03)
```

```
va <- svariog(X=sim03, plot=TRUE)
fit <- fitsvariog(vario=va, ini.cov.pars=c(0.05,4.5), nugget=0.5, max.dist=200)
fit

# graphical display
fit <- fitsvariog(vario=va, ini.cov.pars=c(0.05,4.5), nugget=0.5, max.dist=200, plot=FALSE)
plot(va$svario$u, va$svario$v)
lines(fit$fit)

## fit model to empirical variograms from field data and see how the maximum distance
## to be used can change the results
data(crypho)
va <- svariog(X=crypho, plot=TRUE, messages=FALSE)
fit1 <- fitsvariog(vario=va, ini.cov.pars=c(0.03,100), nugget=0.1, max.dist=300, plot = TRUE)
fit2 <- fitsvariog(vario=va, ini.cov.pars=c(0.03,100), nugget=0.1, max.dist=600, plot = TRUE)

# plot results
plot(va$svario$u, va$svario$v)
lines(fit1$fit, col="blue")
lines(fit2$fit, col="red")
```

---

gene2geo — *Creates a* `ggene` *object from a* `genind` *object and a coordinates data frame*

Description

The function creates a ggene object from a `genind` object and a data frame containing the coordinates.

Usage

```
gene2geo(X, coord)
```

Arguments

`X`	A `genind` object (see package `adegenet`) giving the genetic data of a set of individuals.
`coord`	A data frame containing the coordinates of the individuals.

Details

Genetic data are contained in the `genind` object. Such objects can be created from various file formats (genepop, genetix...) using the package `adegenet`.

Value

An object of class ggene with 5 items:

`tab`	Data frame of the dummy variable coding for the presence of each allele. The number of rows is the number of individuals, the number of columns is the total number of alleles (all locus pooled).

`coord`	The x and y cordinates (longitude and latitude).
`nloc`	Number of different locus.
`loc`	The number of different alleles per locus.
`locnames`	The names of the different locus.

Warning

This function is intended be used to manage diploid data ONLY. Haploid data are not supported by `gene2geno` and should be handled using `tab2geo`. Caution is needed as regards missing data (NAs). NAs must be removed or replaced prior to analysing the dataset. One option developped in the package `adegenet` is to replace NAs by the NAs by the mean allele frequency. `ggene` has no function to handle NAs in raw data and users are referred to the package adegenet and its function `scaleGen`. This function will allow processing diploid data as a `genind` object. For haploid datasets, NAs removal must be done directly by users.

Author(s)

Jean-Pierre Rossi <ggene.package@gmail.com>

See Also

`tab2geo`

Examples

```
library(adegenet)
dat <- read.genepop(system.file("extdata/sim_03.gen",package="ggene"), ncode=3)
xy <- read.csv(system.file("extdata/xysim_01.csv",package="ggene"),header=FALSE)[1:dim(dat$tab)[1],]
data <- gene2geo(X=dat, coord=xy)
class(data)
str(data)
```

`genocount` *Identifies the different genotypes in a data set*

Description

Identifies the different genotypes in a ggene object. Returns a vector indicating the identity of each genotype.

Usage

```
genocount(X)
```

Arguments

`X`	a ggene object.

Value

`vec`	the list of genotypes.
`n`	the number of different genotypes in the data set.

Author(s)

Jean-Pierre Rossi <ggene.package@gmail.com>

Examples

```
data(Wagner)
count <- genocount(X=Wagner)
count

data(crypho)

#compute the weights
count <- genocount(X=crypho)
mat <- genoweight(X=crypho,genotyp=count$vec)

#compute the weighted variogram
wva <- varioWeight(X=crypho, weights=mat)

#performs the randomizations on weighted variogram
env <- randsvariog(var=wva, X=crypho, nsim=9, bounds=NULL, save.sim=FALSE, weights=mat)
```

`genoweight` *Compute a matrix of weights accounting for recurrent genotypes*

Description

Create a matrix of weights accounting for recurrent genotypes following Wagner et al (2005).

Usage

```
genoweight(X, genotypes)
```

Arguments

`X`	a ggene object.
`genotypes`	A vector giving genotype identity. Typically produced by `genocount`.

Details

The weights are computed as the inverse of the number of similar genotypes following the proposition of Wagner et al (2005).

Value

A `dist` object corresponding to the matrix of weights for recurent genotypes. Each individual receives a weight corresponding to the inverse of the number of similar genotypes.

Note

The first example below illustrates the use of `genoweight` with the example given in Wagner et al. 2005 page 1752.

Author(s)

Jean-Pierre Rossi <ggene.package@gmail.com>

References

Wagner, H. H., R. Holderegger, S. Werth, F. Gugerli, S. E. Hoebee and C. Scheidegger. 2005. Variogram analysis of the spatial genetic structure of continuous populations using multilocus microsatellite data. Genetics 169, 1739-1752.

Examples

```
data(Wagner)
count <- genocount(X=Wagner) ;  count
mat <- genoweight(X=Wagner,genotyp=count$vec) ; mat

data(crypho)
#compute the weights
count <- genocount(X=crypho)
mat <- genoweight(X=crypho,genotyp=count$vec)

#compute the weighted variogram
wva <- varioWeight(X=crypho, weights=mat)

## Not run:
#performs the randomizations on weighted variogram
env <- randsvariog(var=wva, X=crypho, nsim=9, bounds=NULL, save.sim=FALSE, weights=mat)
## End(Not run)
```

`larix1350` *A dataset corresponding to the microsatellite data for a set of 189 European larch trees* Larix decidua *sampled in the French Alps at the altitude 1350 m asl*

Description

A set of 13 locus for 189 individuals of European larch (*Larix decidua*) sampled in an experimental plot at the altitude of 1350 m asl near the village of Villar-Saint-Pancrace (Hautes-Alpes, France). 10 individuals were removed from the published dataset because they showed some null alleles.

Usage

```
data(larix1350)
```

Format

An object of class ggene.

Source

Nardin, M., Guerin, V., Musch, B., Rousselle, Y., Sanchez, L., Rossi, J.-P., Gerber, S., Paques, L., Rozenberg, P. 2015. Genetic differentiation of European larch along an altitudinal gradient in the French Alps. Annals of Forest Science 72, 517-527.

Examples

```
data(larix1350)

# check sampling scheme
plot(larix1350$coord[,1],larix1350$coord[,2], asp=1)

# compute variogram
va <- svariog(X=larix1350, uvec=distlag(dist=larix1350$coord, dmin=0, distance.lag=3),
 plot=FALSE)
plot(va$svario$u, va$svario$v)

## Not run:
# compute statistical envelope
env <- randsvariog(var=va, X=larix1350, nsim=30, bounds=c(0.025, 0.975), save.sim=FALSE)

# plot results
plot(env$svario$u, env$svario$v, ylim=range(env$env), xlab="distance", ylab="semi-variance")
points(env$svario$u, env$env[,1], type="l")
points(env$svario$u, env$env[,2], type="l")
## End(Not run)
```

`larix2300` *A dataset corresponding to the microsatellite data for a set of 175 European larch trees* Larix decidua *sampled in the French Alps at the altitude of 2300 m asl*

Description

A set of 13 locus for 165 individuals of European larch (*Larix decidua*) sampled in an experimental plot at the altitude of 2300 m asl near the village of Villar-Saint-Pancrace (Hautes-Alpes, France).

Usage

```
data(larix2300)
```

Format

An object of class ggene.

Source

Nardin, M., Guerin, V., Musch, B., Rousselle, Y., Sanchez, L., Rossi, J.-P., Gerber, S., Paques, L., Rozenberg, P. 2015. Genetic differentiation of European larch along an altitudinal gradient in the French Alps. Annals of Forest Science 72, 517-527.

Examples

```
data(larix2300)

# check sampling scheme
plot(larix2300$coord[,1],larix2300$coord[,2], asp=1)

# compute variogram
```

```
va <- svariog(X=larix2300, uvec=distlag(dist=larix2300$coord, dmin=0, distance.lag=3), plot=FALSE)
plot(va$svario$u, va$svario$v)

## Not run:
# compute statistical envelope
env <- randsvariog(var=va, X=larix2300, nsim=30, bounds=c(0.025, 0.975), save.sim=FALSE)

# plot results
plot(env$svario$u, env$svario$v, ylim=range(env$env), xlab="distance", ylab="semivariance")
points(env$svario$u, env$env[,1], type="l")
points(env$svario$u, env$env[,2], type="l")
## End(Not run)

##
# compute directional variograms
d0_225 <- svariog(X=larix2300,direction=0, tolerance=22.5, unit.angle="degrees")
d45_225 <- svariog(X=larix2300,direction=45, tolerance=22.5, unit.angle="degrees")
d90_225 <- svariog(X=larix2300,direction=90, tolerance=22.5, unit.angle="degrees")
d135_225 <- svariog(X=larix2300,direction=135, tolerance=22.5, unit.angle="degrees")

# plot the results
plot(va$svario$u, va$svario$v, type="b", ylim=range(c(va$svario$v,
d0_225$svario$v, d45_225$svario$v, d90_225$svario$v, d135_225$svario$v))
,xlab="distance", ylab="semi-variance")

points(d0_225$svario$u, d0_225$svario$v, type="b", lty=2)
points(d45_225$svario$u, d45_225$svario$v, type="b", col="red", lty=2)
points(d90_225$svario$u, d90_225$svario$v, type="b", col="blue", lty=2)
points(d135_225$svario$u, d135_225$svario$v, type="b", col="green", lty=2)

legend("topleft", legend=c("omnidirectional", expression(0 * degree), expression(45 * degree),
expression(90 * degree), expression(135 * degree)), lty=c(1,2,2,2,2,2),
col=c("black","black","red","blue","green"), bty="n")

# plot semivariance locus by locus
va <- svariog(X=larix2300)
plot(va$svario$u,va$bylocus[[1]]$gamma.by.locus, xlab="distance", ylab="semivariance",
type="n", ylim=c(0,0.5))
cols <- rainbow(length(va$bylocus))

for(i in 1:(length(va$bylocus))){
  points(va$svario$u,va$bylocus[[i]]$gamma.by.locus, type="l", col=cols[i])
  }
legend("bottomleft", legend=larix2300$locnames, col=cols, bty="n", lty=1, ncol=3)

## Not run:
# compute variogram map
map <- svarmap(X=larix2300,cutoff=150, width=5) ; plot(map)
## End(Not run)

###
# fit exponential model to the empirical variogram
fit <- fitsvariog(vario=va, ini.cov.pars=c(0.5,20), nugget=0.2, max.dist=60, plot = FALSE)
fit$param

# plot results
plot(va$svario$u, va$svario$v)
```

```
lines(fit$fit)
```

`randsvariog` *Performs permutations and computes the variogram statistical envelope*

Description

Randomization of the genetic data with no change of the spatial position of the individuals. The function return the statistical envelope for the specified quantiles.

Usage

```
randsvariog(var, X, weights=NULL, nsim=99, bounds=NULL, save.sim=FALSE, ...)
```

Arguments

`var`	An object of class `svariog`, typically an output of the function `svariog`.
`X`	A `ggene` object.
`weights`	A matrix of weights. Typically an object produced by `genoweight`.
`nsim`	The number of permutations to be performed.
`bounds`	2 numerical values indicating the probability for which the quantiles are computed. Defaults set to 0.975 and 0.025.
`save.sim`	A logical value. If `TRUE`, the simulated values are to be returned.
`...`	additional arguments to be passed to the function `variog` (package `geoR`).

Details

The function performs permutations of the genetic data while keeping the spatial position of the individuals unchanged. The semivariance is computed for each "randomized" dataset. The upper and lower bounds are derived from the randomized values of the semivariance at each spatial lag.

Value

`svario`	An object of class `"variogram"` created with `svariog`.
`env`	the quantiles corresponding to the upper and lower bounds;
`simul`	OPTIONAL. A data.frame where columns contain the values of the semi-variance for each lag for each permutation.

Author(s)

Jean-Pierre Rossi <ggene.package@gmail.com>

See Also

`svariog`, `genoweight`

Examples

```
data(larix2300)

# check sampling scheme
plot(larix2300$coord[,1],larix2300$coord[,2])

# compute variogram
va <- svariog(X=larix2300, uvec=distlag(dist=larix2300$coord, dmin=0, distance.lag=3),
plot=FALSE)
plot(va$svario$u, va$svario$v)

## Not run:
# compute statistical envelope
env <- randsvariog(var=va, X=larix2300, nsim=30, bounds=c(0.025, 0.975), save.sim=FALSE)

# plot results
plot(env$svario$u, env$svario$v, ylim=range(env$env), xlab="distance", ylab="semi-variance")
points(env$svario$u, env$env[,1], type="l")
points(env$svario$u, env$env[,2], type="l")

# Repeated genotypes: envelopes for raw and weighted variograms
data(crypho)

#compute the weights
count <- genocount(X=crypho)
mat <- genoweight(X=crypho,genotyp=count$vec)

#performs the randomizations on raw variogram
va <- svariog(X=crypho, plot=FALSE)
env <- randsvariog(var=va, X=crypho, nsim=30, bounds=NULL, save.sim=FALSE)

#compute the weighted variogram
wva <- varioWeight(X=crypho, weights=mat)

#performs the randomizations on weighted variogram
env2 <- randsvariog(var=wva, X=crypho, nsim=30, bounds=NULL, save.sim=FALSE, weights=mat)

# Plot results

# plot results
xx <- c(wva$svario$u, rev(wva$svario$u))
yy <- c(env$env[,1], rev(env$env[,2]))
plot(xx, yy, type = "n", xlab = "distance", ylab = "semivariance",
 ylim=range(c(env$env[,1], env$env[,2], env2$env[,1], env2$env[,2])))
polygon(xx, yy, col = "lightgrey", border = "black")
xx <- c(wva$svario$u, rev(wva$svario$u))
yy <- c(env2$env[,1], env2$env[,2])
points(xx, yy, type = "l")
polygon(xx, yy, col = "lightblue", border = "blue")

points(wva$svario$u, wva$svario$v, col="blue", typ="b")
points(wva$svario$u, wva$svario$gamma, col="black", type="b", lty="solid", bty="n")

legend("top", legend=c("raw", "weighted"), col=c("black", "blue"), lty="solid", bty="n")
```

```
## End(Not run)
```

---

sim01 *A haploid genotypic data set simulated with the software IBDsim (Leblois et al 2009)*

Description

Simulated data under SMM generated using the software IBDsim (Leblois et al 2009), 625 gene copies, 20 loci, 25 x 25 haploid individuals evolving at G=0 on a 300 x 300 lattice with absorbing boundaries, mutation proba=0.001, Mrca Moy= 286842, MRCA MAX=617227.

Usage

```
data(sim01)
```

Format

An object of class ggene.

Source

Leblois, R., A. Estoup and F. Rousset 2009. IBDSim: a computer program to simulate genotypic data under isolation by distance. Molecular Ecology Resources 9: 107-109.

Examples

```
data(sim01)

# plot the spatial distribution of individuals
plot(sim01$coord[,1], sim01$coord[,2], asp=1)

# compute variogram
va <- svariog(X=sim01, uvec=distlag(dist=sim01$coord, dmin=1, distance.lag=2), plot=FALSE)
plot(va$svario$u, va$svario$v)

# fit exponential model to the empirical variogram
fit <- fitsvariog(vario=va, ini.cov.pars=c(0.5,20), nugget=0.5, max.dist=30, plot = FALSE)
fit$param

# plot results
plot(va$svario$u, va$svario$v)
lines(fit$fit)

## Not run:

# compute statistical envelope
env <- randsvariog(var=va, X=sim01, nsim=30, bounds=c(0.025, 0.975), save.sim=FALSE)
# plot results
plot(env$svario$u, env$svario$v, ylim=range(env$env), xlab="distance", ylab="semi-variance")
points(env$svario$u, env$env[,1], type="l")
points(env$svario$u, env$env[,2], type="l")
```

```
# compute directional variograms
d0_225 <- svariog(X=sim01,direction=0, tolerance=22.5, unit.angle="degrees")
d45_225 <- svariog(X=sim01,direction=45, tolerance=22.5, unit.angle="degrees")
d90_225 <- svariog(X=sim01,direction=90, tolerance=22.5, unit.angle="degrees")
d135_225 <- svariog(X=sim01,direction=135, tolerance=22.5, unit.angle="degrees")

# plot the results
plot(va$svario$u, va$svario$v, type="b", ylim=range(c(va$svario$v, d0_225$svario$v,
d45_225$svario$v, d90_225$svario$v, d135_225$svario$v)) ,xlab="distance",
ylab="semi-variance")

points(d0_225$svario$u, d0_225$svario$v, type="b", lty=2)
points(d45_225$svario$u, d45_225$svario$v, type="b", col="red", lty=2)
points(d90_225$svario$u, d90_225$svario$v, type="b", col="blue", lty=2)
points(d135_225$svario$u, d135_225$svario$v, type="b", col="green", lty=2)

legend("topleft", legend=c("omnidirectional", expression(0 * degree), expression(45 * degree),
expression(90 * degree), expression(135 * degree)), lty=c(1,2,2,2,2,2),
col=c("black","black","red","blue","green"), bty="n")

## End(Not run)
```

sim02 *A haploid genotypic dataset simulated with the software IBDsim (Leblois et al 2009).*

Description

The genetic data of `sim02` correspond to the first 100 individuals of the dataset `sim01` but spatial coordinates have been modified so that the resulting pattern is clumped (See `sim01` for details).

Usage

```
data(sim02)
```

Format

An object of class ggene.

Source

Leblois, R., A. Estoup and F. Rousset 2009. IBDSim: a computer program to simulate genotypic data under isolation by distance. Molecular Ecology Resources 9: 107-109.

Examples

```
data(sim02)

# plot the spatial distribution of individuals
plot(sim02$coord[,1], sim02$coord[,2], asp=1)
```

```
# compute variogram
va <- svariog(X=sim02, uvec=distlag(dist=sim01$coord, dmin=1, distance.lag=1), plot=FALSE)
plot(va$svario$u, va$svario$v, xlab="distance", ylab="semivariance")
abline(h=va$Hhat, col="red", lty="dashed")

## Not run:
  # compute statistical envelope
  env <- randsvariog(var=va, X=sim02, nsim=30, bounds=c(0.025, 0.975), save.sim=FALSE)
  # plot results
 plot(env$svario$u, env$svario$v, ylim=range(env$env), xlab="distance", ylab="semi-variance")
  abline(h=va$Hhat, col="red", lty="dashed")
  points(env$svario$u, env$env[,1], type="l")
  points(env$svario$u, env$env[,2], type="l")

## End(Not run)
```

sim03 *A haploid genotypic data set simulated with the software IBDsim (Leblois et al 2009)*

Description

The genetic data of `sim03` correspond to the first 100 individuals of the data contained in `sim01` (See `sim01` for details).

Usage

```
data(sim03)
```

Format

An object of class ggene.

Source

Leblois, R., A. Estoup and F. Rousset 2009. IBDSim: a computer program to simulate genotypic data under isolation by distance. Molecular Ecology Resources 9: 107-109.

Examples

```
data(sim03)

# plot the spatial distribution of individuals
plot(sim03$coord[,1], sim03$coord[,2], asp=1)

# compute variogram
va <- svariog(X=sim03, uvec=distlag(dist=sim03$coord, dmin=1, distance.lag=1), plot=FALSE)
plot(va$svario$u, va$svario$v, xlab="distance", ylab="semivariance")
abline(h=va$Hhat, col="red", lty="dashed")

## Not run:
  # compute statistical envelope
  env <- randsvariog(var=va, X=sim03, nsim=30, bounds=c(0.025, 0.975), save.sim=FALSE)
```

```
    # plot results
   plot(env$svario$u, env$svario$v, ylim=range(env$env), xlab="distance", ylab="semi-variance")
    abline(h=va$Hhat, col="red", lty="dashed")
    points(env$svario$u, env$env[,1], type="l")
    points(env$svario$u, env$env[,2], type="l")

## End(Not run)

# fit exponential model to the empirical variogram
fit <- fitsvariog(vario=va, ini.cov.pars=c(0.5,20), nugget=0.8, max.dist=30, plot = FALSE)
fit$param

# plot results
plot(va$svario$u, va$svario$v)
lines(fit$fit)
```

---

subsetdata *Extracts data subsets*

Description

The function diplays the individuals from a ggene object and allows to extract a subset of points by clicking the display.

Usage

```
subsetdata(X, L = NULL, col = NULL)
```

Arguments

`X`	A ggene object.
`L`	An optional list of ggene objects resulting from previous runs of the function (see details and examples).
`col`	The color to be used to display the points.

Details

This function is derived from the function `clickpoly` from package `spatstat` which allows the user to create a polygonal window by interactively clicking on the screen display. `subsetdata` returns the corresponding subset of individuals in the form of a new ggene object. The argument `L` allows superimposing previous extractions to the current graphical display to ease manual polygon definition (see example and package vignette : `vignette("introduction_to_ggene")`).

Value

`X`	A ggene object corresponding tot he extracted data subset.
`subset.ppp`	A ppp object (see package `spatstat` for details) containing the points corresponging to the extracted indivicuals. Mostly useful for graphical outputs and internal use.
`original.ppp`	A ppp `object` corresponding to the original set of individuals location. For internal use or graphical outputs.

Note

See the package vignette for examples, for that type `vignette("introduction_to_ggene")`.

Author(s)

Jean-Pierre Rossi <ggene.package@gmail.com>

Examples

```
## Not run:

data(sim02)
sub <- subsetdata(X=sim02, col="blue")
class(sub[[1]])
sub2 <- subsetdata(X=sim02, col="blue", L=list(sub))

## End(Not run)
```

`svariog` *Semivariogram computation*

Description

Computes empirical variograms. Allows directional variograms estimation and weighting for recurrent genotypes

Usage

```
svariog(X, plot=TRUE, messages=FALSE, ...)
```

Arguments

`X`	a `ggene` object.
`plot`	logical with default to `FALSE`. If `TRUE` a plot of the variogram is displayed.
`messages`	Logical. If `TRUE` the function returns various messages during computation.
`...`	optional arguments see `variog` (package `geoR`).

Details

The function relies on the function `variog` from package `geoR` and accepts similar arguments. Readers are referred to the help page of the function `variog` for details.

The optional arguments are :

uvec a vector with values used to define the variogram binning. Possibly created using `distlag`.

max.dist a numerical value defining the maximum distance considered to constitute pairs of individuals.

direction a numerical value comprised in the interval: $[0, \pi]$ radians ($[0, 180]$ degrees) for the directional (azimuth) angle defining the directional variograms.

tolerance numerical value comprised in the interval $[0, \pi/2]$ radians ($[0, 90]$ degrees) indicating the tolerance angle for directional variograms computation. Default set to $\pi/8$.

unit.angle defines the unit for the specification of angles (`radians` or `degrees`). Default set to `radians`.

Value

An object of class `svariog` A list of 5 items:

`svario`	an object of class `variogram` (package `geoR`).
`Hhat`	conventional estimation of the variance (gene diversity): Hhat in Wagner et al. 2005.
`bylocus`	A list with the semivariance estimated locus by locus
`loc`	A vector indicating the number of alleles by locus
`unit.angle`	the unit for the specification of angles

Author(s)

Jean-Pierre Rossi <ggene.package@gmail.com>

References

Wagner, H. H., R. Holderegger, S. Werth, F. Gugerli, S. E. Hoebee and C. Scheidegger. 2005. Variogram analysis of the spatial genetic structure of continuous populations using multilocus microsatellite data. Genetics 169, 1739-1752.

See Also

`varioWeight`, `svarmap`, `randsvariog`, `fitsvariog`

Examples

```
# omnidirectional variogram: simple computation
data(larix2300)
va <- svariog(X=larix2300, plot=TRUE)

# omnidirectional variogram: changing the distance increment
d <- distlag(dist=larix2300$coord, dmin=0, distance.lag=5)
va <- svariog(X=larix2300, uvec=d, plot=TRUE)

# omnidirectional variogram: changing the distance increment again
d <- distlag(dist=larix2300$coord, dmin=0, distance.lag=10)
va2 <- svariog(X=larix2300, uvec=d, plot=TRUE)

# plotting
plot(va$svario$u, va$svario$v)
points(va2$svario$u, va2$svario$v, col="red", pch=6)

# another example
data(larix1350)
va3 <- svariog(X=larix1350, uvec=distlag(dist=larix1350$coord,
      dmin=0, distance.lag=10), plot=FALSE)
plot(va3$svario$u, va3$svario$v)

## Not run:
# computing and plotting statistical envelopes
env <- randsvariog(var=va3, X=larix1350, nsim=30,
  bounds=c(0.025, 0.975), save.sim=FALSE)
plot(env$svario$u, env$svario$v, ylim=range(env$env),
     xlab="distance", ylab="semi-variance")
points(env$svario$u, env$env[,1], type="l")
points(env$svario$u, env$env[,2], type="l")

## End(Not run)

## weighting for recurrent genotypes
data(crypho)
# compute matrix of weights
count <- genocount(X=crypho)
mat <- genoweight(X=crypho,genotyp=count$vec)
d <- distlag(dist=crypho$coord, dmin=0,distance.lag=50)
# compute variogram
wva <- varioWeight(X=crypho, weights=mat,  uvec=d)
#plot the weighted variogram
plot(wva$svario$u, wva$svario$gamma, col="black", type="b",
     ylim=range(c(wva$svario$gamma,wva$svario$v)))
#add the variogram for raw data
points(wva$svario$u, wva$svario$v, col="red", type="b")

## Computation of the directional variogram
## Not run:
data(aniso)
va <- svariog(X=aniso, plot=TRUE)

d0_225 <- svariog(X=aniso,direction=0, tolerance=22.5, unit.angle="degrees")
d45_225 <- svariog(X=aniso,direction=45, tolerance=22.5,  unit.angle="degrees")
d90_225 <- svariog(X=aniso,direction=90, tolerance=22.5,  unit.angle="degrees")
d135_225 <- svariog(X=aniso,direction=135, tolerance=22.5,  unit.angle="degrees")

plot(va$svario$u, va$svario$v, type="b", ylim=range(c(va$svario$v,d0_225$svario$v,
d45_225$svario$v, d90_225$svario$v, d135_225$svario$v)), xlab="distance",
ylab="semivariance")

points(d0_225$svario$u, d0_225$svario$v, type="b", lty=2)
points(d45_225$svario$u, d45_225$svario$v, type="b", col="red", lty=2)
points(d90_225$svario$u, d90_225$svario$v, type="b", col="blue", lty=2)
points(d135_225$svario$u, d135_225$svario$v, type="b", col="green", lty=2)

legend("topleft", legend=c("omnidirectional", expression(0 * degree), expression(45 * degree),
  expression(90 * degree), expression(135 * degree)), lty=c(1,2,2,2,2,2),
col=c("black", "black","red","blue","green"), bty="n")

## End(Not run)

## omnidirectional variogram: weighting for recurrent genotypes
data(crypho)

# compute matrix of weights
count <- genocount(X=crypho)
mat <- genoweight(X=crypho,genotyp=count$vec)
d <- distlag(dist=crypho$coord, dmin=0,distance.lag=50)

# compute variogram
wva <- varioWeight(X=crypho, weights=mat,  uvec=d)

#plot the weighted variogram
```

```
plot(wva$svario$u, wva$svario$gamma, col="black", type="b",
     ylim=range(c(wva$svario$gamma,wva$svario$v)))

#add the variogram for raw data
points(wva$svario$u, wva$svario$v, col="red", type="b")

## plot semivariance locus by locus
data(larix2300)
va <- svariog(X=larix2300)
plot(va$svario$u,va$bylocus[[1]]$gamma.by.locus, xlab="distance", ylab="semivariance",
     type="n", ylim=c(0,0.5))
cols <- rainbow(length(va$bylocus))

for(i in 1:(length(va$bylocus))){
  points(va$svario$u,va$bylocus[[i]]$gamma.by.locus, type="l", col=cols[i])
}
legend("bottomleft", legend=larix2300$locnames, col=cols, bty="n", lty=1, ncol=3)
```

svarmap *Compute the variogram map for a genetic dataset*

Description

The function computes the variogram map for a genetic dataset and returns a map in the form of an object of class `SpatialPixelsDataFrame`.

Usage

```
svarmap(X, cutoff, width)
```

Arguments

`X`	a ggene object.
`cutoff`	a numerical value indicating the separation distance up to which pairs of individuals are included in semivariance estimates. Default set to the length of the diagonal of the box spanning the data divided by three.
`width`	a numercial value indicating the width of distance intervals into which individuals are grouped for semivariance estimates.

Details

Variogram maps are also referred to as 'variogram surface' (Isaaks and Srivastava, 1989 p. 149). The method is an effective way to search for anisotropy axes. The tolerance on h, the separating vector, is defined in a rectangular coordinate system. Type `vignette("ggene_introduction")` for details and examples. `svarmap` relies on the function `variogram` from the package `gstat`.

Value

`map`	A `SpatialPixelsDataFrame` object [package `sp`]

.

Author(s)

Jean-Pierre Rossi <ggene.package@gmail.com>

References

Isaaks, E. H. and R. M. Srivastava 1989. Applied geostatistics, Oxford University Press.

See Also

svariog

Examples

```
data(aniso)

map <- svarmap(X=aniso,cutoff=20, width=1)
plot(map)
# a very clear anisotropy can be seen along the 45 degrees direction

# compute omnidirectional variogram and the directional variogram in the 45 degrees direction
# for comparison
va <- svariog(X=aniso, plot=FALSE)
d45_225 <- svariog(X=aniso,direction=45, tolerance=22.5,  unit.angle="degrees")

# plot variograms
plot(va$svario$u, va$svario$v, type="b", ylim=range(c(va$svario$v, d45_225$svario$v)),
xlab="distance", ylab="semivariance", lty=2)
points(d45_225$svario$u, d45_225$svario$v, type="b", col="red", lty=2)
legend("bottomright", legend=c("omnidirectional", expression(45 * degree)), lty=c(2,2,2,2,2,2),
       col=c("black", "red"), pch=1, bty="n")

## Not run:
data(crypho)
map <- svarmap(X=crypho,cutoff=500, width=25)
plot(map)
# changing the threshold value i.e. only values computed from a number
# of data pairs >= threshold values are shown
plot(map, threshold = 50)

# changing the width
map <- svarmap(X=crypho,cutoff=500, width=20)
plot(map)

map <- svarmap(X=crypho,cutoff=500, width=40)
plot(map)

# changing cutoff
map <- svarmap(X=crypho, cutoff=250, width=20)
plot(map)

map <- svarmap(X=crypho, cutoff=500, width=20)
plot(map)
plot(map, threshold = 30)

## End(Not run)
```

tab2geo — *Create a* ggene *object from data frames*

Description

The function create a ggene object from a data frame containing the genetic data and a second data frame containing the coordinates.

Usage

```
tab2geo(X, coord)
```

Arguments

`X`	A data frame with the genetic data of a set of individuals. Locus in columns, individuals as rows.
`coord`	A data frame containing the coordinates of the individuals.

Details

`tab2geo` was written to handle haploid data which are not supported in the function `gene2geo`.

Value

An object of class `ggene` with 5 items:

`tab`	Data frame of the dummy variable coding for the presence of each allele. The number of rows is the number of individuals, the number of columns is the total number of alleles (all locus pooled).
`coord`	The x and y cordinates (longitude and latitude).
`nloc`	Number of different locus.
`loc`	The number of different alleles per locus.
`locnames`	The names of the different locus.

Warning

Caution is needed as regards missing data (NAs). NAs must be removed or replaced prior to analysing the dataset. One option developped in the package `adegenet` is to replace NAs by the NAs by the mean allele frequency. `ggene` has no function to handle NAs in raw data and users are referred to the package adegenet and its function `scaleGen`. This function will allow processing diploid data as a `genind` object. For haploid datasets, NAs removal must be done directly by users.

Author(s)

Jean-Pierre Rossi <ggene.package@gmail.com>

See Also

`tab2geo`

Examples

```
dat <- read.csv(system.file("extdata/sim_01.csv", package="ggene"),
 header=FALSE)
xy <- read.csv(system.file("extdata/xysim_01.csv", package="ggene"),
 header=FALSE)
data <- tab2geo(X=dat, coord=xy)
class(data)
str(data)
```

`varioWeight` *Variogram computation with weighting for recurrent genotypes*

Description

Compute the omnidirectional variogram for genetic dataset while accounting for recurrent genotypes using a weighting matrix.

Usage

```
varioWeight(X, weights, return.mat=FALSE,...)
```

Arguments

`X`	a ggene object
`weights`	A matrix of weights. Typically an object produced by `genoweight`.
`return.mat`	Logical, if `TRUE`, the function returns the matrix of weights.
`...`	arguments to be passed to the function `svariog`..

Details

The function relies on the function `variog` from package `geoR` and accepts similar arguments.

Value

A list with 2 items:

`svario`	an object of class `svariog`
`gamma`	a `dist` object corresponding to the matrix of semi-variance (gene diversity)
`weight`	the matrix of weights

Author(s)

Jean-Pierre Rossi <ggene.package@gmail.com>

References

Wagner, H. H., R. Holderegger, S. Werth, F. Gugerli, S. E. Hoebee and C. Scheidegger. 2005. Variogram analysis of the spatial genetic structure of continuous populations using multilocus microsatellite data. Genetics 169, 1739-1752.

See Also

genocount, genoweight, svariog

Examples

```
data(crypho)

# check sampling scheme
plot(crypho$coord[,1],crypho$coord[,2], asp=1)

# compute matrix of weights
count <- genocount(X=crypho)
mat <- genoweight(X=crypho,genotyp=count$vec)

# compute distance intervals
d <- distlag(dist=crypho$coord, dmin=0,distance.lag=50)

# compute weighted variogram
wva <- varioWeight(X=crypho, weights=mat,  uvec=d)

# plot the variogram for raw data
plot(wva$svario$u, wva$svario$gamma, col="black", type="b",
ylim=range(c(wva$svario$gamma,wva$svario$v)), xlab="distance", ylab="semivariance")

# add the weighted variogram
points(wva$svario$u, wva$svario$v, col="red", type="b", pch=4)

legend("top", legend=c("raw", "weighted"), col=c("black", "red"), lty="solid", pch=c(1,4), bty="n")

## changing distance increment
d <- distlag(dist=crypho$coord, dmin=0,distance.lag=75)
wva <- varioWeight(X=crypho, weights=mat,  uvec=d)

d <- distlag(dist=crypho$coord, dmin=0,distance.lag=50)
wva2 <- varioWeight(X=crypho, weights=mat,  uvec=d)

d <- distlag(dist=crypho$coord, dmin=0,distance.lag=22)
wva3 <- varioWeight(X=crypho, weights=mat,  uvec=d)

plot(wva$svario$u, wva$svario$v, type="b", pch=1, lty=2, xlab="distance (m)",
ylab="semivariance", ylim=range(c(wva$svario$v,wva2$svario$v,wva3$svario$v)))
points(wva2$svario$u, wva2$svario$v, type="b", pch=17, lty=2)
points(wva3$svario$u, wva3$svario$v, type="b", pch=4, lty=2)
title("weighted variograms for different distance increments")
legend("topleft", legend=c("75m", "50m", "25m"), lty=0, pch=c(1,17,4), bty="n")

## Not run:

#performs randomization on raw variogram
va <- svariog(X=crypho, plot=FALSE)
env <- randsvariog(var=va, X=crypho, nsim=30, bounds=NULL, save.sim=FALSE)

#compute the weighted variogram
wva <- varioWeight(X=crypho, weights=mat)
```

```
#performs the randomizations on weighted variogram
env2 <- randsvariog(var=wva, X=crypho, nsim=30, bounds=NULL, save.sim=FALSE, weights=mat)

# plot results
xx <- c(wva$svario$u, rev(wva$svario$u))
yy <- c(env$env[,1], rev(env$env[,2]))
plot(xx, yy, type = "n", xlab = "distance", ylab = "semivariance",
 ylim=range(c(env$env[,1], env$env[,2], env2$env[,1], env2$env[,2])))
polygon(xx, yy, col = "lightgrey", border = "black")
xx <- c(wva$svario$u, rev(wva$svario$u))
yy <- c(env2$env[,1], env2$env[,2])
points(xx, yy, type = "l")
polygon(xx, yy, col = "lightblue", border = "blue")
points(wva$svario$u, wva$svario$v, col="blue", typ="b")
points(wva$svario$u, wva$svario$gamma, col="black", type="b", lty="solid", bty="n")


## End(Not run)
```

Wagner *The example data set used in Wagner et al (2005)*

Description

A toy data set used in Wagner al at (2005) p. 1750

Usage

```
data(Wagner)
```

Format

An object of class ggene.

Note

Using this dataset with `varioWeight` and `svariog` leads to various warning messages because there are some co-locatted individuals. This has no effect upon results.

Source

Wagner, H. H., R. Holderegger, S. Werth, F. Gugerli, S. E. Hoebee and C. Scheidegger. 2005. Variogram analysis of the spatial genetic structure of continuous populations using multilocus microsatellite data. Genetics 169, 1739-1752.

Examples

```
data(Wagner)

count <- genocount(X=Wagner)
count
mat <- genoweight(X=Wagner,genotypes=count$vec)
mat
```

```
# NB: individuals 1 and 2 are similar, the weight of this couple is 0

# compute variogram for genetic diversity
wa <- varioWeight(X=Wagner, weights=mat, uvec=c(1,2,3))
# NB: the third distance class corresponding to a lag of 3 distance units is
# omitted here because it involves only one data pair
wa$svario$u # corresponds distance r in Wagner et al 2005 p 1751
wa$svario$gamma # raw semivariances corrsponding to Hhat(r) in Wagner et al 2005 p 1751
wa$svario$n # corresponds distance nr in Wagner et al 2005 p 1751
wa$svario$v # is the weighted semivariance for geneotypic diversity not to be mistaken
# for the values reported for the molecular variance in Wagner et al 2005 p 1752
```

Index