Structural analysis of genetic data using the R package ggene

Jean-Pierre Rossi & Carole Kerdelhué
CBGP - INRA Montpellier http://www6.montpellier.inra.fr/cbgp
ggene.package@gmail.com

For ggene version 1.0.2

2016-06-07

Contents

1 Forwords

This document is an attempt at giving a quick overview of the `R` package `ggene`. The package was designed to provide tools to analyse microsatellite data recorded for geolocated individuals. `ggene` is based on spatially explicit statistical methods and mostly relies on geostatistics *i.e.* variography. The package allows variogram analysis following Wagner et al. (2005) and provides additional tools such as variogram maps. Readers should refer to the numerous textbooks dedicated to geostatistics amongst which are *Geostatistics for natural resources evaluation* (Goovaerts 1997) and *Applied geostatistics* (Isaaks and Srivastava 1989). For an introduction to geostatitsics in `R`, readers should consult *Model-based geostatistics* (Diggle and Ribeiro 2007).

2 Package Overview

2.1 Disclaimer

`ggene` is delivered as it is and without any warranty. If you find a bug, or if you have a suggestion to improve the package, please contact us at ggene.package@gmail.com.

2.2 Funding

The INRA department EFPA provided financial support to a very early version of the package `ggene`. We further benefited from the financial support of the INRA métaprogramme SMaCH (Sustainable Management of Crop Health) through the project COPACABANA.

2.3 Package functions

The table below lists the main functions available from `ggene`.

name	topic	description
`tab2geo`	data input	creates a `ggene` object (useful for haploid organisms)
`gene2geo`	data input	creates a `ggene` object from a `genind` object
`subsetdata`	data management	subsets a `ggene` object by point-and-click
`distlag`	utility	creates customized distance intervals
`genocount`	utility	computes the number of different genotypes in a dataset
`genoweight`	utility	computes the weights associated to the different genotypes
`svariog`	analysis	variography
`svarmap`	analysis	variography
`varioWeight`	analysis	variography
`randsvariog`	analysis	variography
`fitsvariog`	analysis	variography

As usual in the `R` environment, users can access a specific help page by entering `?name of the function` in the R console. Typing

```
?ggene
```

provides an overview of the main package features.

2.4 Datasets available from ggene

Readers are referred to a dedicated vignette entitled `ggene_datasets` accessible by typing `vignette("ggene_datasets")` in the R console.

3 Data management

3.1 Data inputs

Data are imported and converted to an object of class `ggene` using 2 functions, namely `gene2geo` and `tab2geo`.

- `gene2geo` uses an object of `genind` class from the package `adegenet` (Jombart 2008) and a data frame containing the spatial coordinates of the corresponding individuals.
- `tab2geo` operates in a similar way but requires a simple data frame for the genetic data instead of a `genind` object. `tab2geo` is intended to handle haploid data which cannot be easily stored as a `genind` object (see the manual of `adegenet`). Note that missing data can be handled using the function `scaleGen` available in the package `adegenet`.

The chunk of code below shows how to read both coordinates and microsatellite data for an haploid dataset.

```
# read genetic data
sim <- read.csv(system.file("extdata/sim_01.csv", package="ggene"),
                header=FALSE)
# read spatial coordinates
xy.sim <- read.csv(system.file("extdata/xysim_01.csv", package="ggene"),
                   header=FALSE)
# create a ggene object
dat.sim <- tab2geo(X=sim, coord=xy.sim)
```

```
## The number of individuals is  625
## The number of locus is  20
```

```r
str(dat.sim)
```

```
## List of 5
##  $ tab     : num [1:625, 1:502] 0 0 0 0 0 0 0 0 0 0 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:625] "1" "2" "3" "4" ...
##   .. ..$ : chr [1:502] "V1.14" "V1.16" "V1.17" "V1.18" ...
##  $ coord   :'data.frame':    625 obs. of  2 variables:
##   ..$ V1: int [1:625] 150 150 150 150 150 150 150 150 150 150 ...
##   ..$ V2: int [1:625] 150 151 152 153 154 155 156 157 158 159 ...
##  $ nloc    : int 20
##  $ loc     : Named int [1:20] 23 31 33 31 15 36 29 19 25 20 ...
##   ..- attr(*, "names")= chr [1:20] "V1" "V2" "V3" "V4" ...
##  $ locnames: chr [1:20] "V1" "V2" "V3" "V4" ...
##  - attr(*, "class")= chr "ggene"
```

The function `gene2geo` operates the same way but relies on a `genind` object. `genind` objects are created with the package `adegenet` from various data input formats such as `genepop` or `genetix` (see the documentation of the package for details). **Note that ggene comes with no function allowing to read data directly from files.**

Here is an example where we firstly read the genetic data from a file in `genepop` format :

```r
library(adegenet)
dat <- read.genepop(system.file("extdata/sim_03.gen", package="ggene"),
    ncode = 3)
```

```
##
##  Converting data from a Genepop .gen file to a genind object...
##
##
## File description:  Simulated data
##
## ...done.
```

The resulting object has various characteristics:

```r
dat
```

```
## /// GENIND OBJECT /////////
##
##  // 625 individuals; 20 loci; 502 alleles; size: 1.3 Mb
##
```

```
##  // Basic content
##    @tab:  625 x 502 matrix of allele counts
##    @loc.n.all: number of alleles per locus (range: 15-36)
##    @loc.fac: locus factor for the 502 columns of @tab
##    @all.names: list of allele names for each locus
##    @ploidy: ploidy of each individual  (range: 2-2)
##    @type:  codom
##    @call: read.genepop(file = system.file("extdata/sim_03.gen", package = "ggene"),
##     ncode = 3)
##
##  // Optional content
##    @pop: population of each individual (group size range: 625-625)
```

There are 625 individuals.

Let's now read the geographic coordinates :

```
xy <- read.csv(system.file("extdata/xysim_01.csv", package="ggene"),
               header=FALSE)
dim(xy)
```

```
## [1] 625   2
```

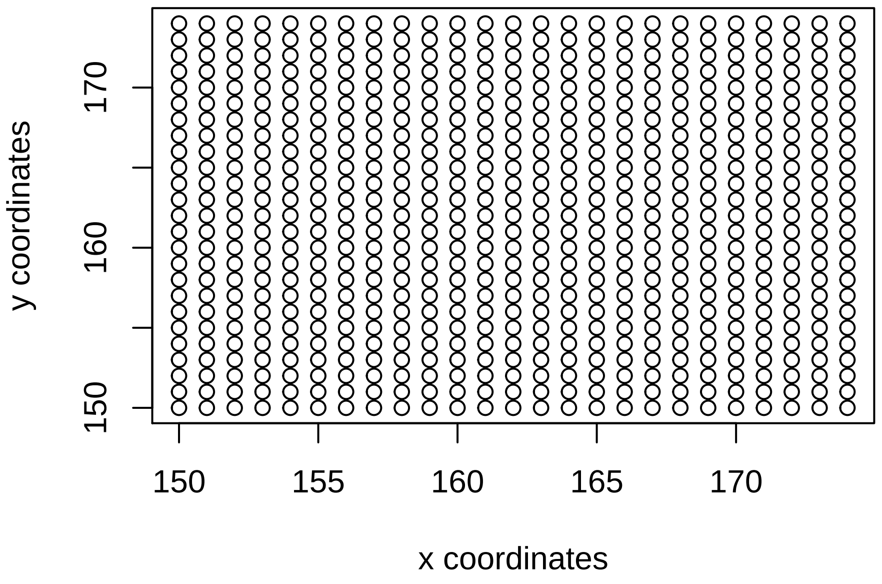
The function `head` displays the first records. There are 2 columns corresponding to abscissa and ordinates, x and y. It is important to check this information since the coordinate file may contain point labels or additional coordinate systems and other information. It is important that users select the correct coordinate system and for that reason, checking the content of the data file is necessary. In addition, it is assumed that coordinates are provided in cartesian system rather that in a longitude/latitude system. **Readers are referred to Bivand et al. (2008) for details.**

```
head(xy)
```

```
##    V1  V2
## 1 150 150
## 2 150 151
## 3 150 152
## 4 150 153
## 5 150 154
## 6 150 155
```

At this point, we can plot the spatial distribution of individuals and visualise the sampling scheme:

```r
plot(xy[,1],xy[,2], xlab="x coordinates", ylab="y coordinates")
```



We can create a `ggene` object using `gene2geo`:

```r
library(ggene)
data <- gene2geo(X=dat, coord=xy)
```

```
## The number of individuals is  625
## The number of locus is  20
```

```r
class(data)
```

```
## [1] "list"  "ggene"
```

3.2 Data subsets

In some cases, one may wish to analyse a subset of the dataset *e.g.* one patch of individuals. The function `subsetdata` allows to extract subsets of a dataset using a polygon created by point-and-click in the display.

```r
data(sim02)
sub <- subsetdata(X=sim02, col="blue")
to add points: click left mouse button in window
       to exit: click middle mouse button
 [The last point should NOT repeat the first point]
```

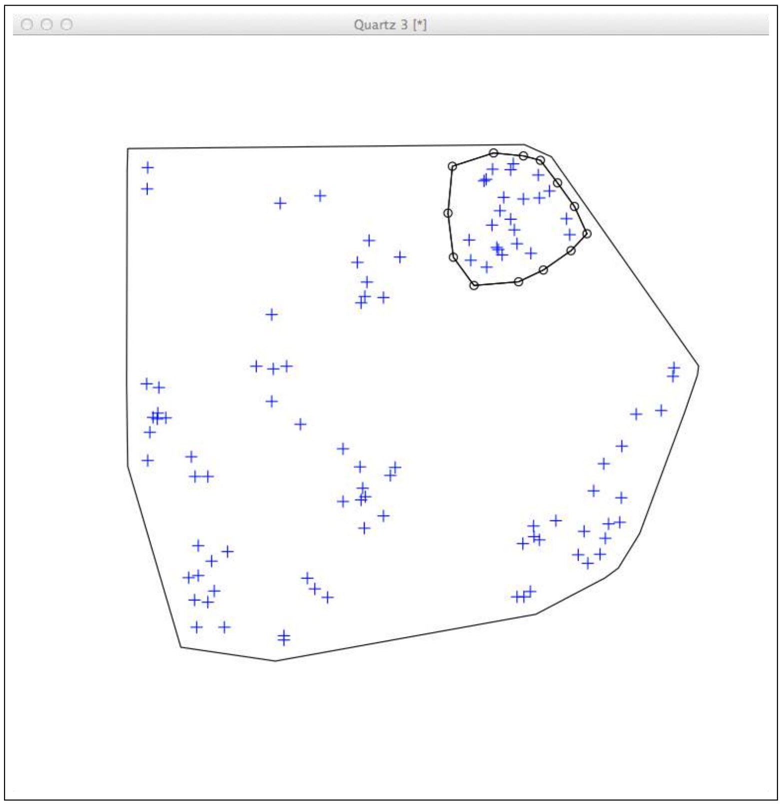


Figure 1: Subsetting a dataset using the function `subsetdata`: screenshot of the polygon definition.

A plot of the individuals location is displayed and user is invited to create a polygon defining the group of points to be extracted by clicking in the display (figure 1 page 8).

The resulting object has 4 slots, the first one being the ggene object corresponding to the subset of data.

```
class(sub[[1]])
[1] "ggene"
```

Sequential extractions are possible by launching the function with previous extracts as a list of arguments (L).

```
> sub2 <- subsetdata(X=sim02, col="blue", L=list(sub))
to add points: click left mouse button in window
       to exit: click middle mouse button
 [The last point should NOT repeat the first point]
```

The definition of the second polygon is done while the first polygon (sub) appears in the plot (figure 2 page 10).

3.3 Superimposed individuals

Several individuals may be collected at the same site *i.e.* their coordinates are similar. This situation is not uncommon in biology while it is unlikely in geology and soil science where a given spatial location supports only one sample. Duplicated coordinates may also point out an error in the data file. For that reason, ggene will issue a warning message when a data set contains points/individuals with duplicated coordinates. This occurs when you read the raw datasets by means of functions tab2geo or gene2geo.

For instance :

```
data(sim02)
# forcing 3 duplicated sample locations
sim02$coord[5,] <- sim02$coord[10,] <- sim02$coord[20,]
# create a new ggene object with duplicate coordinates
sim02bis <- tab2geo(X=sim02$tab, coord=sim02$coord)
```

```
## WARNING: There are some duplicated coordinates.
## You should consider jittering the duplicated coordinates before
## running the anisotropic variography.
## You can use jitterDupCoords from package geoR
## The number of individuals is  100
## The number of locus is  395
```

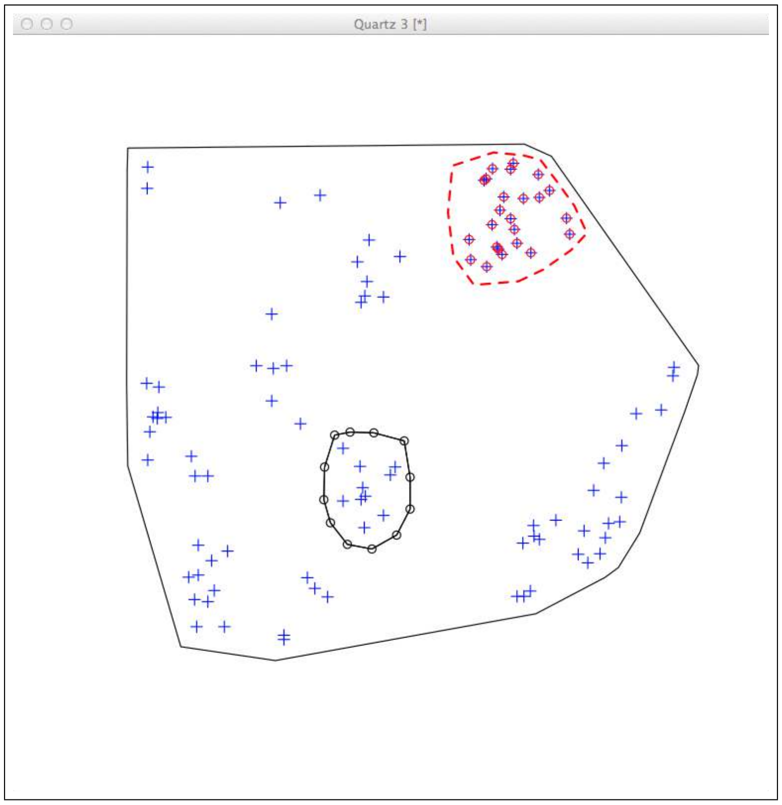


Figure 2: Subsetting a dataset using the function `subsetdata`: screenshot of a second polygon definition. Note that the first polygon `"sub"` appears in the plot to facilitate the positioning a the new polygon.

Duplicated coordinates cause no problem in variogram analysis except for directional variograms. The function `svariog` will issue a warning message when data contain duplicated coordinates but will run correctly in the case of omnidirectional variograms (in that case a bin will be added at the origin: see below). If you want to avoid any warning you can jitter point coordinates. This consists in slightly changing the coordinates and is of no consequence on the analysis outputs provided that the changes remain slight. Jittering the duplicated coordinates can be achieved using `jitterDupCoords` from package `geoR`.

The following code chunk tests for the presence of duplicated coordinates:

```
library(geoR)
# test for duplicated coordinates
dup.coords(sim02bis$coord)
```

```
##      [,1]
## [1,] "5"
## [2,] "10"
## [3,] "20"
## attr(,"class")
## [1] "duplicated.coords"
```

There are 3 duplicated coordinates.

```
#Jitter coordinates
coordbis <- jitter2d(sim02$coord[,], max=0.01)

# test for duplicated coordinates
dup.coords(coordbis$coord)
```

```
## NULL
```

The coordinates have been slightly changed, resulting in no duplicate. Users should consult the help page of the function `jitter2d` and carefully evaluate the value to be given to the argument `max`.

3.4 Managing recurrent genotypes

The presence of repeated genotypes potentially affects the spatial structure and it may be necessary to account for this effect (H. Wagner et al. 2005; Werth S. et al. 2006; Dutech et al. 2008). This can be achieved by using a weighting matrix during the computations (see H. Wagner et al. 2005).

`genocount` and `genoweight` are utilities for preparing such matrices of weights.

- `genocount` identifies and counts the repeated genotypes in a given dataset. The function returns a list of genotypes and a number that identifies each of them in the dataset. The output of `genocount` can be useful on its own but is primarily intended to feed the function `genoweight`.
- `genoweight` computes the weighting matrix following Wagner et al. (2005) using the output of `genocount`.

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

```
## $vec
## [1] 1 1 2 3 3 3
##
## $n
## [1] 3
```

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

```
##           1         2         3         4         5
## 2 0.0000000
## 3 0.5000000 0.5000000
## 4 0.1666667 0.1666667 0.3333333
## 5 0.1666667 0.1666667 0.3333333 0.0000000
## 6 0.1666667 0.1666667 0.3333333 0.0000000 0.0000000
```

The matrix of weights `mat` in the example above can be used to compute the weighted variogram using the function `varioWeight` (see details § 5.2.3).

3.5 Missing data (NAs)

Missing data *i.e.* nuls alleles are not allowed and should be managed prior to data analysis. This can be achieved by removing either individuals and/or locus exhibiting missing data. Another solution is offered by the function `scaleGene` for the package `adegenet`. `scaleGene` allows to replace NAs by the mean allele frequency or by zero. In the present document and in all the examples provided in `ggene` we have removed the individuals exhibiting missing values.

4 Variograms in a nutshell

4.1 Semivariance & the variogram

In geostatistics, spatial structures are usually assessed using the average dissimilarity between data separated by a vector h. The dissimilarity is measured by the *semivariance* which is computed as half the average squared difference between the components of each data pair:

$$\hat{\gamma}(h) = \frac{1}{2n_h} \sum_{a \neq b} \chi_{ab}^{(h)} (x_a - x_b)^2 \tag{1}$$

where x_a and x_b are the values of the observed variable at sites a and b, h is the distance class, n_h the number of individuals separated by a distance h and $\chi_{ab}^{(h)}$ is the Kronecker weight. The Kronecker weight for a data pair ab takes the value $\chi_{ab}^{(h)} = 1$ if the pair belongs to the distance interval h and $\chi_{ab}^{(h)} = 0$ otherwise. In this way, only the pairs of individuals (a, b) within the distance class h are taken into account in the calculation of $\hat{\gamma}(h)$.

The more alike are the individuals separated by h, the smaller $\hat{\gamma}(h)$ and *vice versa*. The plot of $\hat{\gamma}(h)$ against the separating distance h is referred to as the *semi-variogram* or *variogram* for short. It represents the average rate of change of the similarity with distance. Its shape describes the pattern of spatial variation in terms of general form, scales and magnitude. In many cases, $\hat{\gamma}(h)$ increases with h and reaches a plateau roughly equal to the sample variance. The distance h for which $\hat{\gamma}(h)$ reaches the plateau is called the range of the variogram and corresponds to the distance at which individuals/sampling points can be considered as statistically independent (Goovaerts 1997; Isaaks and Srivastava 1989).

The features of the empirical variogram are conveniently summarized by fitting a theoretical model to the estimates of $\hat{\gamma}(h)$. Not all models that seem to fit can be used and only positive definite models are valid (*aka* authorized functions). This question is beyond the scope of the present introduction and readers are referred to geostatistical textbooks such as Isaaks & Srivastava (1989) or Goovaerts (1997). Because positive definiteness is somewhat difficult to check, geostatisticians traditionally use a set of authorized functions that are known to meet the assumption of positive definiteness. Authorized functions are described in various textbooks and can be combined to describe more complex variograms (Goovaerts 1997).

A mentioned above, under certain circumstances, the variogram levels off for a certain distance, referred to as the range b. When h reaches b the semivariance value is maximum and remains constant. Such variograms are referred to as bounded variograms. The plateau is called the *sill* in the geostatistical jargon and corresponds to the variation picked up by the sampling scheme. The *sill* variance is the sum of two terms corresponding to the part of the variance that is explained by the spatial structure of the variable (referred to as C_1, the *spatial* variance or *partial sill*) and an additional term accounting for residual (non spatial) variance referred to as nugget variance C_0 (a term derived from the gold mining jargon)(figure 3 page 14). The nugget variance C_0 encapsulates various sources of variability that are perceived as being spatially independent, *i.e.* that are not autocorrelated at the scale of the study. When there is no spatial structure, the variogram is horizontal and $\hat{\gamma}(h) = C_0$. Such variograms are referred to as 100% nugget models.

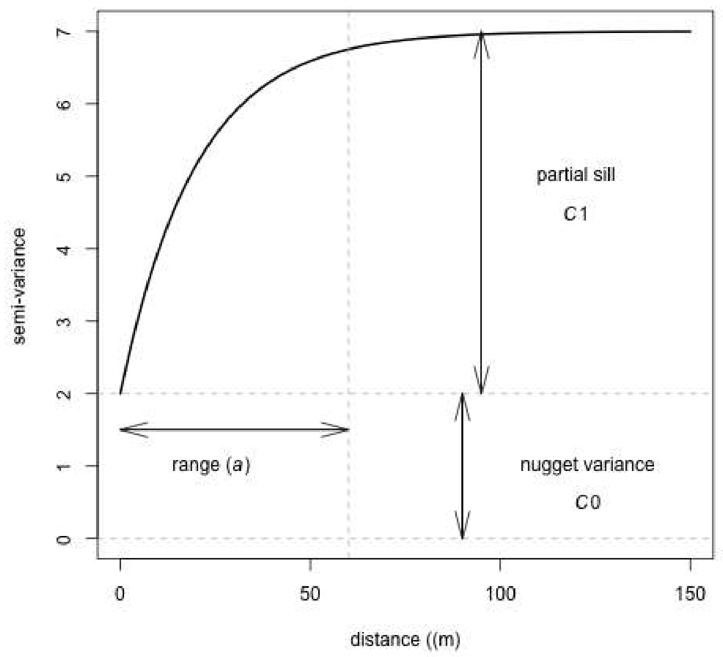


Figure 3: Characteristic features of the variogram: range (a), nugget variance (C_0) and partial sill (C_1).

Fitting theoretical variogram to empirical values has given rise to an important literature and is achieved by using generalized least-squares fitting (Goovaerts 1997) and maximum likelihood (*e.g.* Akaike information criteria Webseter and Oliver (1990)). In practice, theoretical variogram models are fitted to empirical variograms and the model parameters furnish estimates of the partial sill, the range (a) and the *nugget* variance (C_0). Variogram parameters are directly interpretable in terms of strength and extent of the spatial structure. For example, the ratio of the term C_1 to the sum $C_0 + C_1$ provides a measure of the proportion of variance that is spatially dependent. The *range*, b, indicates the separating geographic distance beyond which observations are statistically independent. This is an important issue when designing sampling schemes (Webster and Oliver 1990).

4.2 The exponential model

Some authorized functions are commonly encountered in the geostatistical literature *e.g.* spherical, gaussian or exponential models. We will only consider here the exponential model (H. Wagner et al. 2005). It is defined by :

$$\gamma(h) = C\left(1 - \exp(\frac{-h}{r})\right) \tag{2}$$

where C is the asymptote (sill) and r a distance parameter controlling the spatial extent of the function. In practice, a nugget term C_0 is added to the equation when theoretical models are fitted to empirical variograms.

Exponential models are bounded (Goovaerts 1997) and approach their *sill* asymptotically hence there is no *range* strictly speaking. For practical purposes, however, a effective range $b = 3r$ at which $\gamma(a) = 0.95C$ is defined (Webster and Oliver 1990).

4.3 Variogram of gene diversity

The analysis of spatial genetic structure (SGS) has hitherto mostly involved correlograms and Mantel tests (Hardy and Vekemans 1999) or geostatistical analysis of allelic frequencies (Le Corre et al. 1998). Geostatistical exploration of genetic diversity is rare (Dutech et al. 2008; Werth S. et al. 2006; H. Wagner et al. 2005) and limited to haploid organisms.

Wagner et al. (2005) showed how variograms could be related to gene diversity so as to express SGS of continuous populations. If we consider a set of k dummy variables z_k for k alleles with $z_{ka} = 1$, $z_{ka} = 0.5$ or $z_{ka} = 0$ if individual a is homozygous, heterozygous or does not bear allele k, respectively. The proportion of unlike joins between individuals equals the sum of the variograms of the dummy variables. Wagner et al. (2005) showed that the variogram of multilocus gene diversity $\hat{H}$ (or expected heterozygosity) can be defined as:

$$\hat{H}(h) = \sum_{l=1}^{L} \omega_l \hat{\gamma}_l(h) = \sum_{a \neq b} \sum_{l=1}^{L} \sum_{k} \frac{\omega_l \chi_{ab}^{(h)}}{2n_h} \left(z_{lka} - z_{lkb}\right)^2 \tag{3}$$

where z_{lka} and z_{lkb} are the dummy variables for locus l allele k for individuals a and b, $\chi_{ab}^{(h)}$ is the Kronecker weight, $\omega_l = \frac{1}{L}$ is the weight of locus l with L the number of locus and n_h the number of data pairs separated by a distance of h.

4.4 Isolation by distance and the exponential variogram model

Under isolation–by–distance (*i.e.* limited gene dispersal) in a two-dimensional space, theoretical models predict that kinship between individuals (as well as pairwise F_{st}) decreases approximately linearly with the logarithm of the separating distance (Hardy and Vekemans 1999). The probability of identity of two neutral genes separated by a distance r, $Q(r)$ depends on a gene dispersal function, the mutation rate (μ) and the population effective density D under the assumption of an isotropic dispersal and at drift–dispersal–mutation equilibrium (Vekemans and Hardy 2004). Interestingly, $Q(r)$ is approximately linearly related to $ln(r)$ (in two dimensions) at a rate proportional to $1/D\sigma^2$ for r values ranging between σ and $0.5\sigma/\sqrt{2\mu}$ with σ^2, half the average squared axial parent–offspring distance.

The classical approach (Vekemans and Hardy 2004) consists of regressing the relatedness coefficient $F(r)$ on $log(r)$ and estimating the slope $\hat{b}_F$ which depicts the SGS. This estimation must be done for r ranging between σ and $0.5\sigma/\sqrt{2\mu}$. An additional index, the Sp statistic, has been proposed to characterize the SGS (Vekemans and Hardy 2004). It is:

$$S_p = \frac{b_F}{(1 - F_N)} \tag{4}$$

where F_N is the kinship of immediate neighbors that compete for the same resources. F_N can be estimated by $F(1)$, the value of $F(r)$ for the first distance class.

The S_p statistic is directly related to dispersion parameters (under former hypotheses of IBD and equilibrium)

$$S_p = \frac{1}{4\pi D\sigma^2} \tag{5}$$

Whereas S_p is usually estimated using Moran's I correlogram, the exponential variogram model fitted to empirical values of $\hat{\gamma}(h)$ provides alternate estimates of these statistics (H. Wagner et al. 2005). F_N can be estimated by the ratio of the term C_1 to the *sill* $(C_0 + C_1)$:

$$\hat{F}_N = \frac{C_1}{C_0 + C_1} \tag{6}$$

If an estimate of $F(1)$ is required, the exponential variogram model must be fitted under the constraint of having a fixed nugget parameter $C_0 = \hat{\gamma}(h = 1)$.

The slope parameter b_F is closely related to the range parameter of the variogram. Because the exponential model reaches its *sill* asymptotically, the effective range must be used to estimate b_F (H. Wagner et al. 2005):

$$b = \frac{-3}{\hat{b}_F} \tag{7}$$

Exponential model parameters allow to estimate of the index S_p and hence to the estimation of the dispersal parameter known as the average axial parent–offspring distance.

4.5 What does `ggene` actually compute?

The package provides different functions allowing the estimation of the variogram and a function (`fitsvariog`) dedicated to exponential model fitting to empirical variograms. It provides an estimate of the parameters listed above. `ggene` also allows to compute statistical envelope of the variogram on the basis of random permutations of the genetic data. Two functions provide tools to examine anisotropy. In addition, the package offers some functions allowing to compute the omnidirectional variogram while accounting for the presence of recurrent genotypes following the approach proposed in Wagner et al. (2005).

5 En route for variography

"If you're going through hell, keep going."

Winston Churchill.

5.1 Distance interval to be considered in variography

5.1.1 Distance lags, maximum distance and number of data pairs

The first step to variography is selecting *ad hoc* distance classes. This means choosing a distance lag (also called distance interval or lag width) and a maximum distance value to be considered. Individuals separated by a distance larger than this threshold won't be evaluated. Large distance intervals will thus tend to smooth out local details while small distance intervals may obscure the underlying signal (if any) with too much local "noise". The maximum distance to be accounted for is another important aspect of empirical variograms (Webster and Oliver 1990). Values larger than 1/3 or 1/2 of the maximum inter-sample distance are generally the maximum considered (Journel and Huijbregts 1978; Isaaks and Srivastava 1989; Webster and Oliver 1990) but this is mostly because empirical variograms are used with the aim of fitting a theoretical model ultimately used for interpolation purposes named kriging in geostatistics (Goovaerts 1997). This is not our aim when we study genetic variation and the whole variogram is of interest. It should be noted, however, that the largest (or lowest) distance classes may involve only a few data couples which means that the estimate of the semivariance is poor. The number of data pairs involved in semivariance computation highly depends on the sampling scheme. It must be noted that in genetic survey sampling scheme can be very dependent on natural population spatial pattern while in geology or soil science researchers often use regular schemes. Caution is therefore needed and it is good practice to check the number of data pairs and possibly change the distance interval accordingly. Finally, the sampling scheme is obviously determinant in the distribution of the information along distance classes and therefore each analysis shall be devised according to sampling scheme. Homogeneous sampling schemes are to be preferred and in case of uneven distribution of individuals, users are advised to check the number of data pairs involved in each semivariance estimation. In environmental sciences, geostatistics are often used to analyse data collected following a regular sampling scheme. One advantage is that the number of data pairs per lags decreases more regularly when the distance lag is increased. Random or clumped schemes requires that users be very cautious when interpreting the variogram, particularly for large (small) intersample distances.

5.1.2 The function `distlag`

The function `distlag` allows to quickly build a vector showing the midpoint of the distance classes to be used in variography *i.e.* study of the variogram. `distlag` uses either a set of individual coordinates or an inter-individual euclidean distance matrix. Note that the coordinates (hence the distances derived from) must be defined in a plane. This means that longitude and latitude data, which are recorded in degrees and associated with an ellipse, shall be expressed according to the adequate coordinate reference system (CRS) and projected onto a plane. Readers are referred to the book of Bivand et al. (2008, chap. 4) that provides precious details regarding this point. `distlag` operates with 3 arguments and allow customizing the resulting distance intervals: `dmin` indicates the minimum distance to be accounted for, `distance.lag` indicates the intersample lag distance and `dist.lag.max` denotes the maximum intersample distance to be considered. The function returns a vector that can be passed to other functions of the package such as `svariog` for computation of the semi-variogram.

```
data(aniso)
distlag(dist=aniso$coord, dmin=0, distance.lag=2, dist.lag.max=NULL)
```

```
##  [1]  1  3  5  7  9 11 13 15 17 19 21 23 25 27
```

By default, `distlag` considers all the possible distance classes. It is sometimes useful, however, to restrict the range of the distance classes to get a better view of the spatial structure near the origin. This point is important for instance when the zone of influence of the structure is short with regards to the size of the study area and/or if a model must be fitted very accurately for the very first lags (*i.e.* lower than the variogram range).

```
distlag(dist=aniso$coord, dmin=0, distance.lag=2, dist.lag.max=20)
```

```
##  [1]  1  3  5  7  9 11 13 15 17 19
```

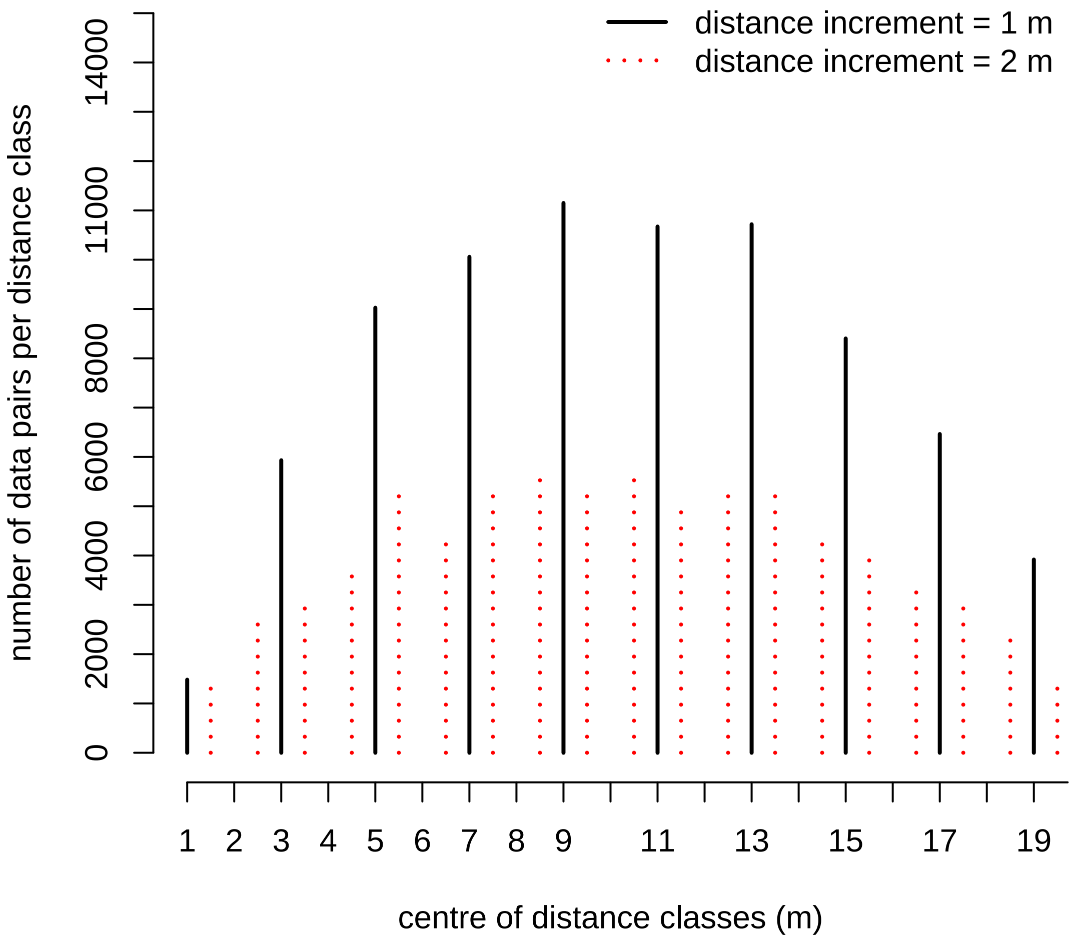
Changing the width of the distance increment has a strong impact on the number of data pairs involved in computations. This can in turn affect the semivariance estimates. The following example shows how increasing the width from 1 meters to 2 meters affects the number of pairs of individuals involved in the semivariance estimations.

```
d <- distlag(dist=aniso$coord, dmin=0, distance.lag=1, dist.lag.max=20)
d
```

```
##  [1]  0.5  1.5  2.5  3.5  4.5  5.5  6.5  7.5  8.5  9.5 10.5 11.5 12.5 13.5
## [15] 14.5 15.5 16.5 17.5 18.5 19.5
```

```
d1 <- distlag(dist=aniso$coord, dmin=0, distance.lag=2, dist.lag.max=20)
d1
```

```
##  [1]  1  3  5  7  9 11 13 15 17 19
```



We shall return to this issue when dealing with variogram computation and model fitting.

5.2 Computing omnidirectional empirical variogram

As would be expected, the *pièce de résistance* of the package `ggene` is its functions allowing variography. `ggene` has 3 functions dedicated to semivariance computation: `svariog`, `varioWeight` and `svarmap`.

- `svariog` computes the variogram for the gene diversity (termed $\hat{H}(h)$ in Wagner et al. (2005)). The function also returns the locus-by-locus semivariance and the conventional estimator of the gene diversity $\hat{H}(h)$. It also implements directional variograms (see below).
- `varioWeight` computes omnidirectional variograms and allows accounting for clonality following the weighting scheme proposed in Wagner et al. (2005). We saw earlier how the functions `genocount` and `genoweight` could be used to compute weighting matrix which, in turn, can feed the function `varioWeight` to compute the weigthed variogram.

- `svarmap` computes the variogram map *aka* variogram surface (Isaaks and Srivastava 1989, 149). The method, although rarely used even in the field of geology where it was originally developed, is an effective way to search for anisotropy axes.

5.2.1 The *Larix decidua* example

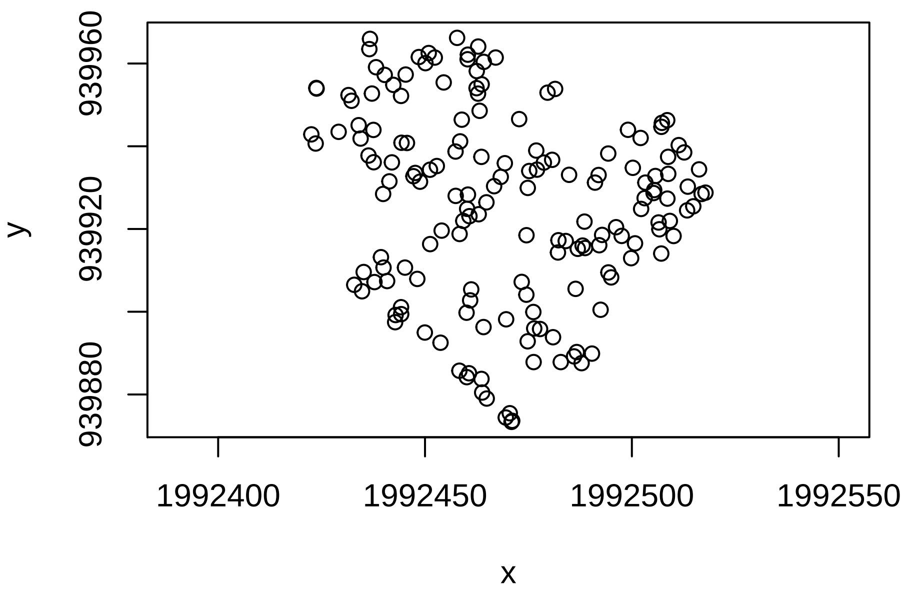
5.2.1.1 The sampling scheme and the distance lag

As seen before, omnidirectional variograms are computed without accounting for directional variation. We will start by examining the example dataset called `larix2300` which provides a set of 13 micosatellite locus recorded for 175 individual trees located in an experimental plot at an altitude of 2300 m asl. in the French Alps (Nardin et al. 2015).

```
data(larix2300)
```

A quick examination of their spatial distribution reveals that the trees are spatially clumped. Such a pattern originates from various ecological processes and deserves a study in its own right (Rossi et al. 2014).

```
plot(larix2300$coord[,1],larix2300$coord[,2], asp=1, xlab="x", ylab="y")
```



Note that the argument `asp` was set to 1 when calling the function `plot`. `asp` controls the `y/x` aspect ratio and a value of 1 ensures that one data unit in the `x` direction is equal in length to one data unit in the `y` direction (see `plot.default`).

Because trees form small patches, many individuals are clumped and a sizeable amount of genetic information occurs at short spatial scales. We therefore want to explore the spatial genetic variation with small separating lags because if the lags were too large, local features might be smoothed out.

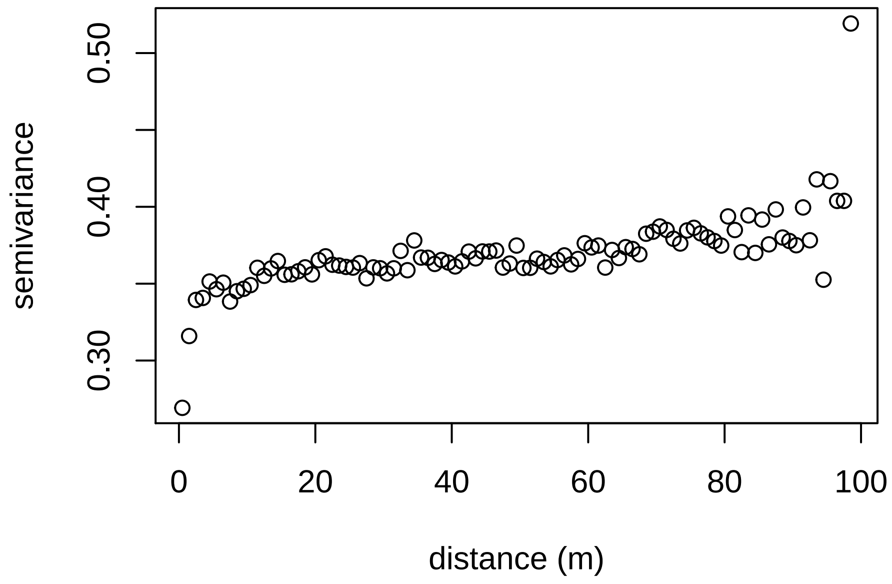
Again, `distlag` allows to customize the distance intervals:

```
d <- distlag(dist=larix2300$coord, dmin=0, distance.lag=1)
d
```

```
##  [1]  0.5  1.5  2.5  3.5  4.5  5.5  6.5  7.5  8.5  9.5 10.5 11.5 12.5 13.5
## [15] 14.5 15.5 16.5 17.5 18.5 19.5 20.5 21.5 22.5 23.5 24.5 25.5 26.5 27.5
## [29] 28.5 29.5 30.5 31.5 32.5 33.5 34.5 35.5 36.5 37.5 38.5 39.5 40.5 41.5
## [43] 42.5 43.5 44.5 45.5 46.5 47.5 48.5 49.5 50.5 51.5 52.5 53.5 54.5 55.5
## [57] 56.5 57.5 58.5 59.5 60.5 61.5 62.5 63.5 64.5 65.5 66.5 67.5 68.5 69.5
## [71] 70.5 71.5 72.5 73.5 74.5 75.5 76.5 77.5 78.5 79.5 80.5 81.5 82.5 83.5
## [85] 84.5 85.5 86.5 87.5 88.5 89.5 90.5 91.5 92.5 93.5 94.5 95.5 96.5 97.5
## [99] 98.5
```

We now estimate the variogram using `d` as the separating distance:

```
va <- svariog(X=larix2300, uvec=d, plot=FALSE)
plot(va$svario$u, va$svario$v, type="p", xlab="distance (m)",
     ylab="semivariance")
```



Note that when `plot=TRUE` a basic variogram is plotted. **From this plot, the main features of the variogram can be seen.**

5.2.1.2 General shape of the empirical variograms

The semivariance increases with increasing inter-individual distance until it reaches a plateau and remains more or less constant. At very large distances (*e.g.* >15 m in our case) the gene diversity may fluctuate as the result of sampling effects: the number of data pairs (tree couples involved in the semivariance estimation) is often much smaller for large separating distances. Obviously, this depends on the spatial distribution of individuals. In the case of the `larix` dataset, almost all trees have been investigated and as such the sampling scheme largeley conveys the natural distribution of the trees which is clumped.

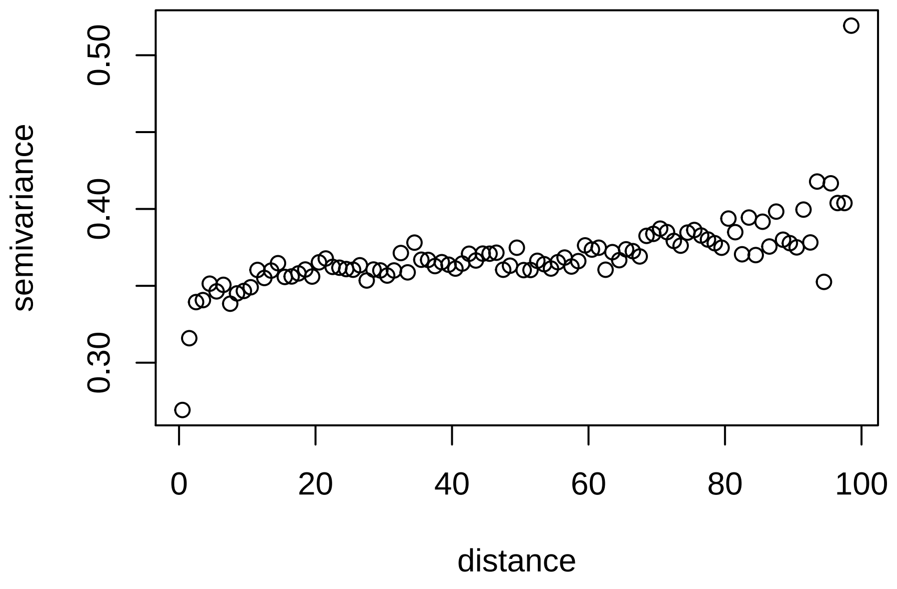
5.2.1.3 A glance at the scale(s): the variogram range(s)

A very interesting feature of the variogram is the distance at which it reaches a plateau. It is referred to as the range of the variogram and provides an information about the spatial scale at which the structure occurs. Individuals separated by a distance lower than the range are considered as statistically independent. Below that range, individuals are more similar than expected by chance. Consequently, if one wants to set up a sampling scheme where individuals are expected to be independent, they must be separated by a distance larger than or equal to the range. This has been discussed in details in various papers and text books (Oliver 2010; Webster1985; Webster and Oliver 1990)

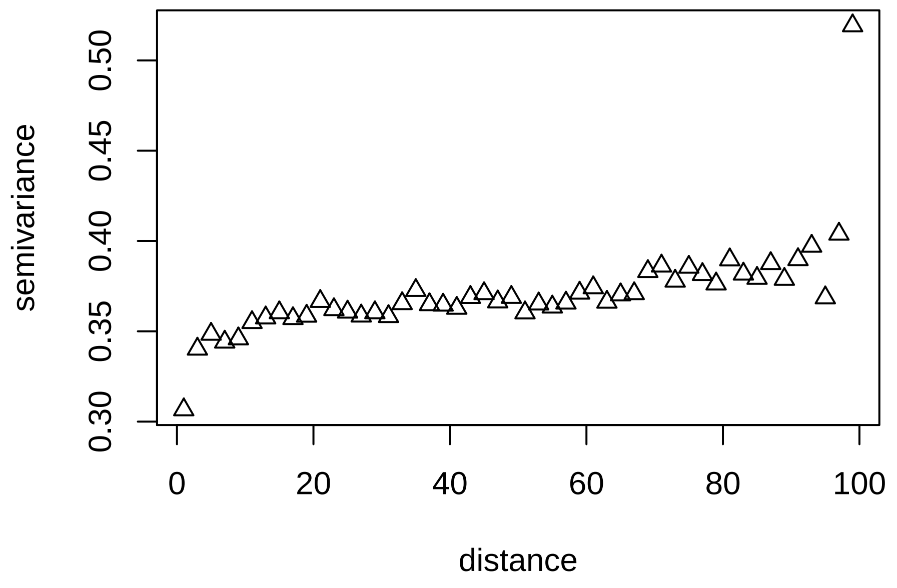
5.2.1.4 Playing with the lag distance

Increasing the lag increment smooths out local variability which is sometimes very useful to get a better picture of the global pattern. Various trials are generally necessary to get a clear view of different scales at which the spatial variation occurs. In the case of `larix2300` we can increase the lag distance:

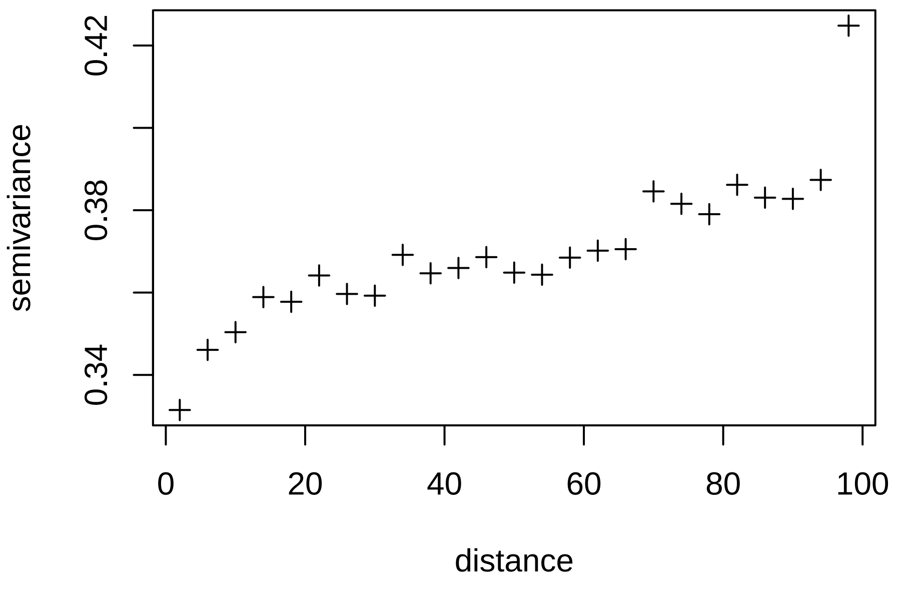
```r
va1 <- svariog(X=larix2300, uvec=distlag(dist=larix2300$coord, dmin=0,
                                         distance.lag=1), plot=FALSE)
va2 <- svariog(X=larix2300, uvec=distlag(dist=larix2300$coord, dmin=0,
                                         distance.lag=2), plot=FALSE)
va4 <- svariog(X=larix2300, uvec=distlag(dist=larix2300$coord, dmin=0,
                                         distance.lag=4), plot=FALSE)
va8 <- svariog(X=larix2300, uvec=distlag(dist=larix2300$coord, dmin=0,
                                         distance.lag=8), plot=FALSE)

plot(va1$svario$u, va1$svario$v, xlab="distance", ylab="semivariance")
```



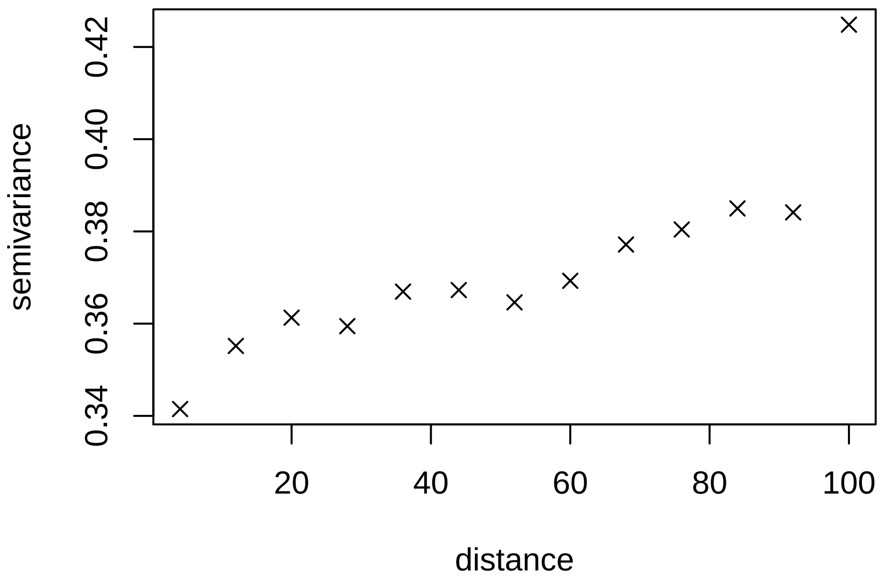
```r
plot(va2$svario$u, va2$svario$v, pch=2, xlab="distance", ylab="semivariance")
```



```r
plot(va4$svario$u, va4$svario$v, pch=3, xlab="distance", ylab="semivariance")
```



```r
plot(va8$svario$u, va8$svario$v, pch=4, xlab="distance", ylab="semivariance")
```



Not only larger increments lead to smoother variograms but they also help detect more or less hidden structures. Here, increments of 8 m lead to a variogram exhibiting a clear structure at distances ranging from 0 to *ca.* 60 m **superimposed** upon another structure at larger scales correponding to the increase in semivariance from 60 m. The topic of superimposed structures and how variograms can be of help in their detection and characterization has been adressed in several papers (Burrough 1983; Bellier et al. 2007; Jiménez et al. 2014) and will be delt in more details later. For the moment, we need to make sure that the second phase of semivariance increase is not solely due to sampling effect *i.e.* lower number of data pairs for larger separating distances.

`svario$n` allows checking that the number of data pairs involved is sufficient:

```
va1$svario$n
```

```
##  [1]   7  28  40  60  55  81  73  84 107 114 106 114 142 140 144 138 151
## [18] 155 165 150 169 185 213 198 214 211 191 184 224 179 223 192 220 197
## [35] 230 219 212 219 202 212 207 193 239 214 197 220 225 199 234 225 205
## [52] 193 197 192 176 210 206 176 199 199 176 176 176 174 164 168 145 134
## [69] 145 142 141 116 135 111 107 125 111  98  92  92  78  68  60  53  50
## [86]  38  34  38  25  28  12  18  12  11   9   3   6   3   2
```

```
va8$svario$n
```

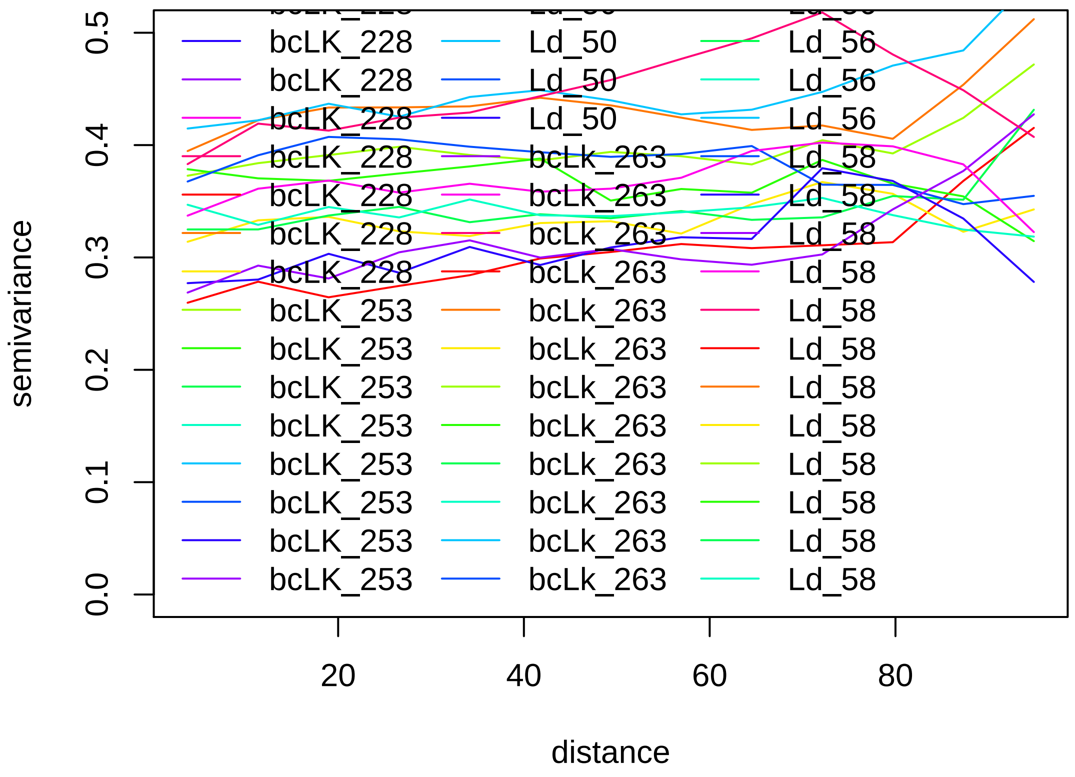
```
##  [1]  428 1005 1386 1618 1711 1694 1632 1482 1155  871  419  118   11
```

A commonly accepted threshold for the minimum number of data pairs is 50 (Journel and Huijbregts 1978). We see that an interval of 1 m leads to a relativeley small number of pairs of individuals both for the lower (1 m) and higher (8 m) separating distances. These results illustrate how distance increment affects our perception of the spatial pattern at hand. It can also noticeably impact model fitting as we will see later.

5.2.1.5 Locus-by-locus variograms

The variogram is the sum of the semivariance estimated for each locus. It may be useful, in some cases, to examine if the spatial signal changes according to the locus considered. This is possible with the slot `$bylocus` returned by the function `svariog` where the semivariance computed for each locus is stored.

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

# 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)
```

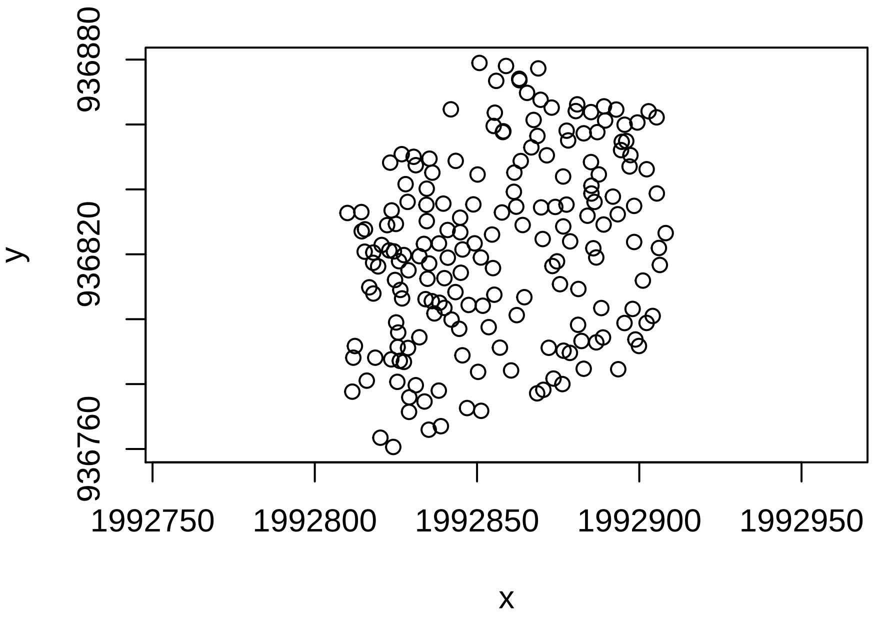


5.2.2 Departure from randomness: more *Larix decidua*!

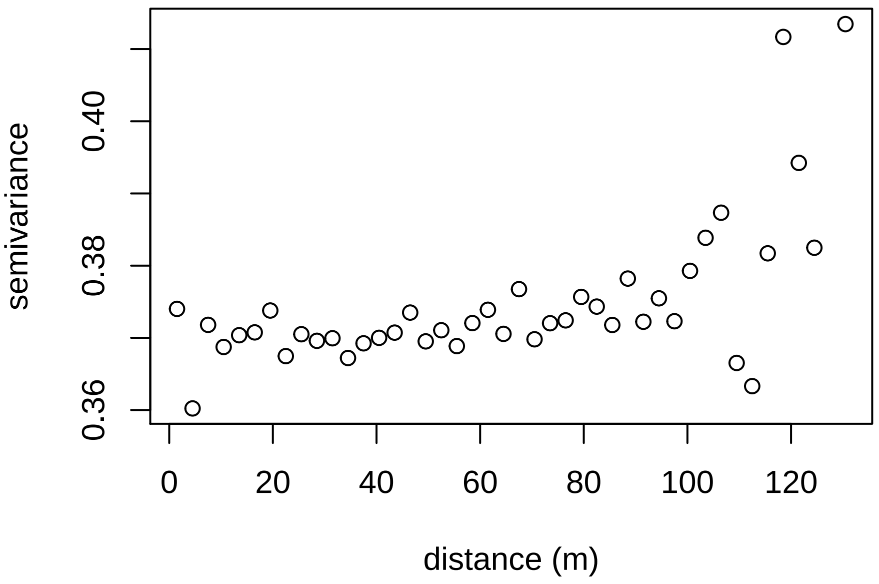
5.2.2.1 Statistical envelopes

`larix1350` provides another example dataset of *Larix decidua* trees. Here, trees were surveyed at lower altitudes (ca.1300 m asl.). Let's see if we can detect a spatial structure within this dataset:

```
data(larix1350)
# examine sampling scheme
plot(larix1350$coord[,1],larix1350$coord[,2], xlab="x", ylab="y", asp=1)
```



```r
va <- svariog(X=larix1350, uvec=distlag(dist=larix1350$coord,
                              dmin=0, distance.lag=3), plot=FALSE)
plot(va$svario$u, va$svario$v, type="p", xlab="distance (m)",
     ylab="semivariance")
```



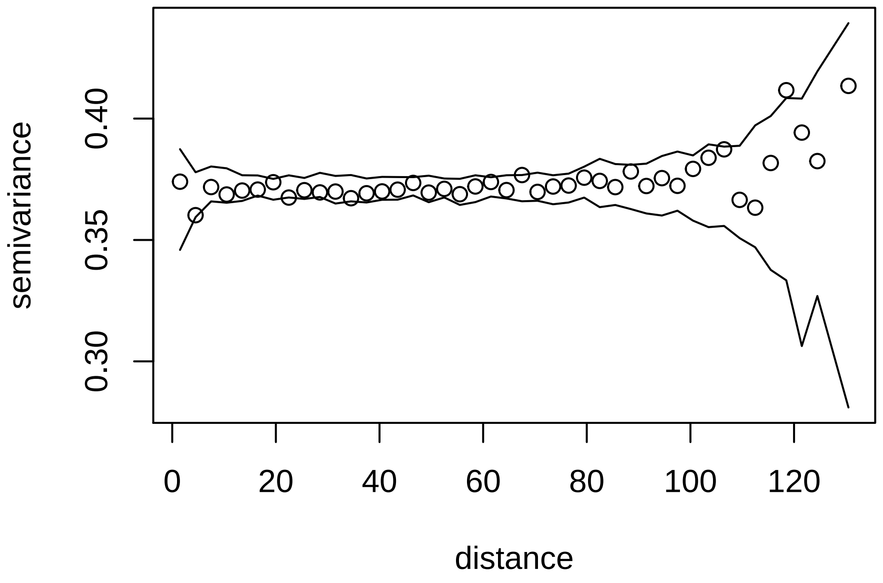
The resulting variogram appears to be somewhat flat, thus indicating that the semivariance (gene diversity) is not changing much with increasing separating distance. Such signal suggests an absence of spatial genetic structure (SGS) and various factors can explain such a lack of structuration (Nardin et al. 2015). The question remains to determine if an empirical variogram is significantly different from what can be expected by chance. `ggene` comes with the function `randsvariog` that performs randomizations of the genotype spatial distribution and recomputes the variogram from the randomized datasets. Readers are referred to Legendre and Legendre (1998) for details about randomization tests in the context of spatial analyses. `randsvariog` randomizes the genetic data associated to each individual. Note that genomes are not changed, only their spatial distribution are permuted between individuals, hence the sampling design is kept unchanged.

```
env <- randsvariog(var=va, X=larix1350, nsim=30, bounds=c(0.025, 0.975),
                   save.sim=FALSE)
```

```
## ..............................
## done
```

The randomizations are used to compute the bounds of the statistical envelope for the observed semivariance. The graphical output is simply obtained with:

```
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")
```

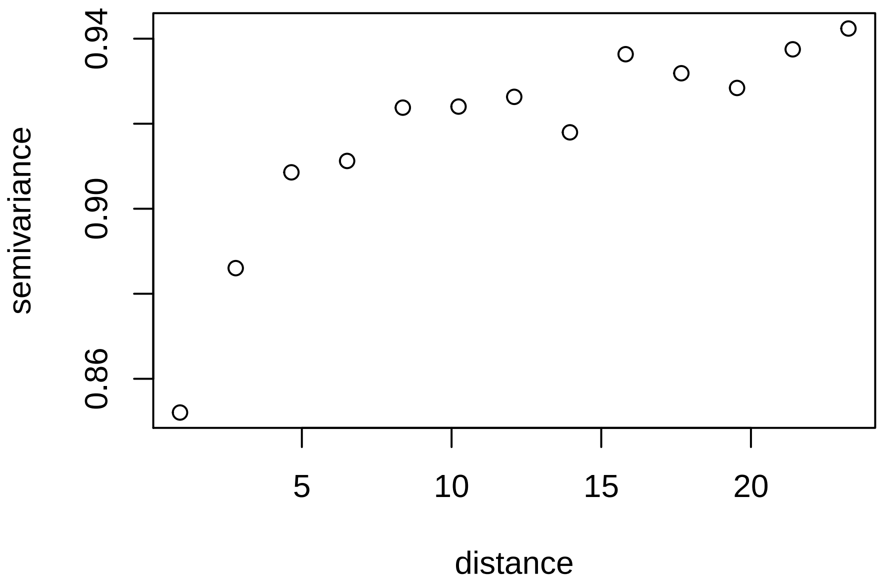


The results show that most of the observed values lie between the the 2.5% and 97.5% bounds suggesting the absence of significant departure from spatial randomness. The number of permutations was kept low in this simple example. More randomizations would be necessary and different tests using various distance increments would be useful to fully investigate the presence of a SGS.

5.2.2.2 Notes on statistical testing

When the argument `save.sim = TRUE`, `randsvariog` returns a `data.frame` with all the simulated semivariances for the different distance classes. The data are provided in the slot `$simul` and they can be used to compute new envelope bounds *i.e.* for new quantile values without redoing the randomizations themselves. For example, the code below computes the variogram for the dataset `sim03` and the corresponding enveloppe for 30 randomizations and the quantile values of 0.025 and 0.975.

```
data(sim03)
var <-svariog(X=sim03, plot=FALSE)
plot(var$svario$u, var$svario$v, xlab="distance", ylab="semivariance")
```



```
env <-randsvariog(var=var, X=sim03, nsim=30, bounds=c(0.025, 0.975),
  save.sim=TRUE)
```

```
## ..............................
## done
```

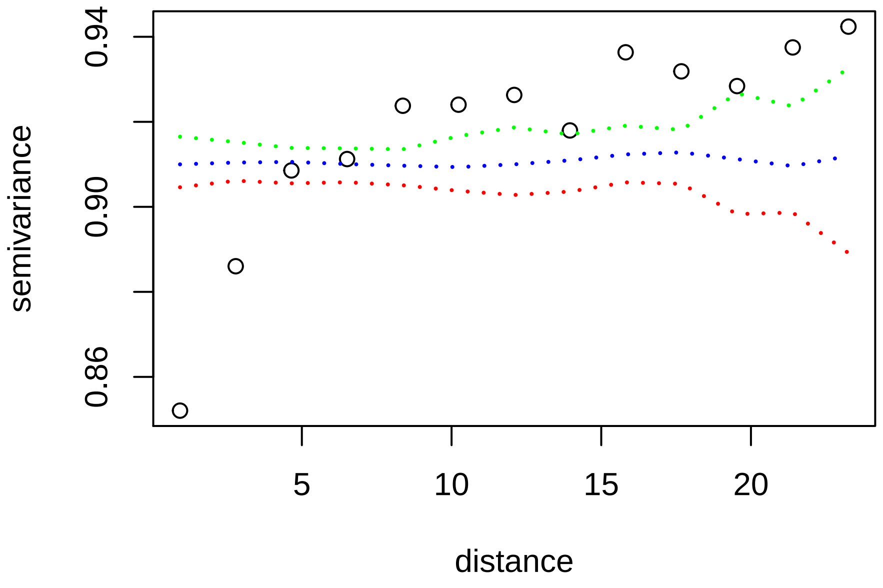
The semivariance after randomization, which mimics the expected values under the null hypothesis of complete spatial randomness, are returned in `env$simul`:

```
dim(env$simul)
```

```
## [1] 13 30
```

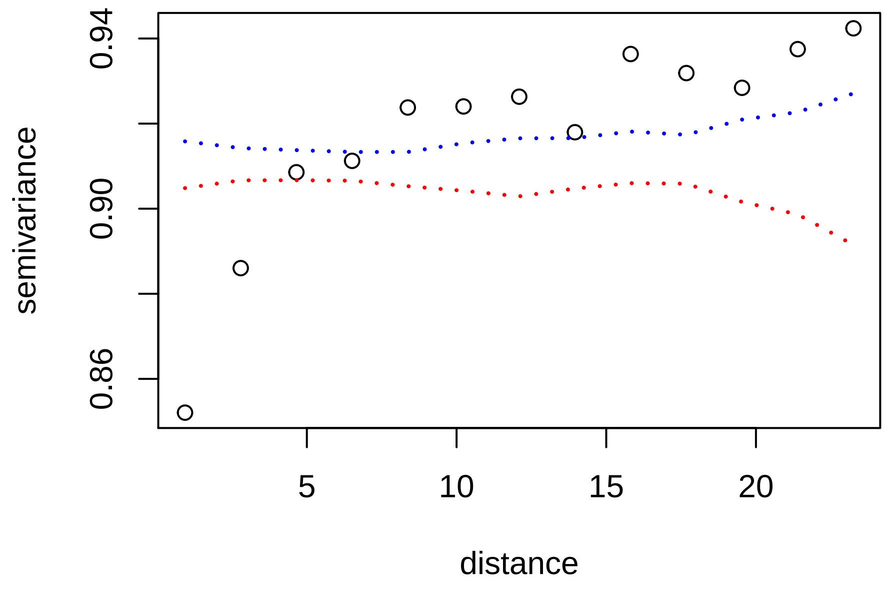
The `data.frame` has a number of rows equal to the number of distance lags and a number of columns corresponding to the number of randomizations. From this `data.frame` we can compute new values describing the variability of the randomized values. Here we compute and plot the minimum, the maximum and the median values:

```
min <- apply(X=env$simul, MARGIN=1, FUN=min)
median <- apply(X=env$simul, MARGIN=1, FUN=median)
max <- apply(X=env$simul, MARGIN=1, FUN=max)

plot(var$svario$u, var$svario$v, xlab="distance", ylab="semivariance")
points(env$svario$u, min, type="l", lty="dotted", col="red", lwd=2)
points(env$svario$u, median, type="l", lty="dotted", col="blue", lwd=2)
points(env$svario$u, max, type="l", lty="dotted", col="green", lwd=2)
```



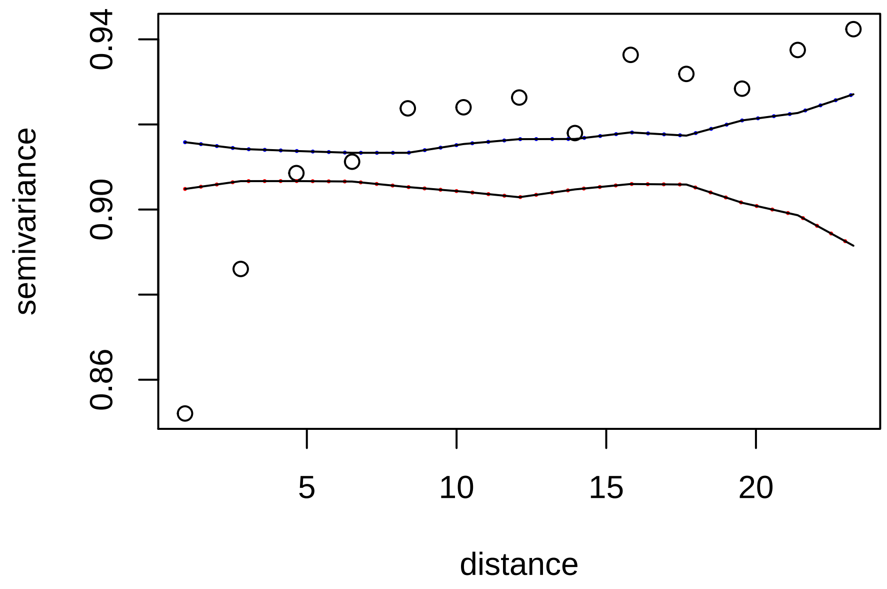
What happens if we select quantile values equal to 0.025 and 0.975 ?

```
q1 <- apply(X=env$simul, MARGIN=1, FUN=quantile, prob=0.025)
q2 <- apply(X=env$simul, MARGIN=1, FUN=quantile, prob=0.975)

plot(var$svario$u, var$svario$v, xlab="distance", ylab="semivariance")
points(env$svario$u, q1, type="l", lty="dotted", col="red", lwd=2)
points(env$svario$u, q2, type="l", lty="dotted", col="blue", lwd=2)
```



We now plot the output of `randsvariog` computed for these quantiles and stored in `$env[]` :

```
plot(var$svario$u, var$svario$v, xlab="distance", ylab="semivariance")
points(env$svario$u, q1, type="l", lty="dotted", col="red", lwd=2)
points(env$svario$u, q2, type="l", lty="dotted", col="blue", lwd=2)

points(env$svario$u, env$env[,1], type="l")
points(env$svario$u, env$env[,2], type="l")
```



The curves are superimposed.

5.2.2.3 Remarks on flat variograms

The variogram for `larix1350` is an example of what is called a *horizontal* variogram. There is no significant change in the semivariance as the average inter-individual distance increases. It means that individuals are independent from a statistical viewpoint. Interestingly, this situation is also observed when a SGS is present but when we consider data couples separated by a distance *larger* than the variogram range. When the empirical variogram appears to be flat, users should try to decrease the lag and examine the behavior of the semivariance for the first distance lags alone to make sure that short-scale patterns are not obscured. The statistical envelopes may also be of help to determine if a structure is present.

5.2.3 Accounting for recurrent genotypes

5.2.3.1 Computing weighted empirical variogram

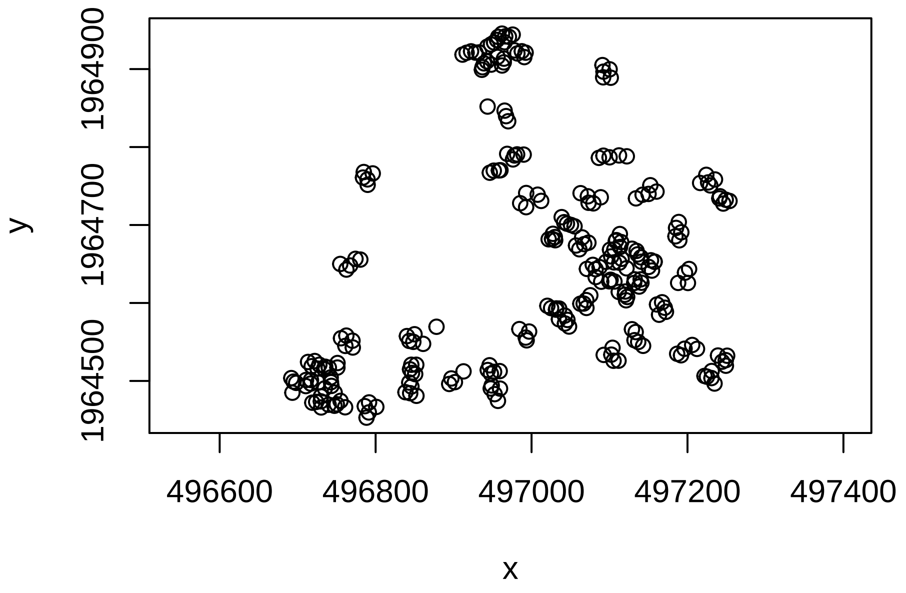
In its present form, `ggene` allows to account for repeated genotypes in omnidirectional variogram analysis alone. The option may become available for directional variograms in the future. As stated by Wagner et al. (2005), the principle of the analysis is to compute the variogram while decreasing the weight given to data couples including individuals whose genotype is repeated. Practically, this is achieved through the computation of a weighting matrix that is passed to the function `varioWeight`. We saw above (§ 3.4) how that so-called matrix of weights is computed using functions `genocount` and `genoweight`.

The function returns a variogram reflecting the SGS after weithing for repeats following the proposal of Wagner et al. (2005). Suppose that a significant SGS is observed for a given dataset where a certain degree of clonality is observed. The weighted variogram indicates if the SGS is solely due to the spatial distribution of repeated genotypes or if it conveys another source of variability.

We illustrate this analysis using a dataset named `crypho` consisting in a set of 10 locus for 276 individuals of the chestnut blight fungus *Cryphonectria parasitica* (CBF) (Dutech et al. 2008). This organism is haploid and a certain level of clonality is observed (Dutech et al. 2008).

We first compute the number of different genotypes using `genocount`:

```r
data(crypho)
# check sampling scheme
plot(crypho$coord[,1],crypho$coord[,2], xlab="x", ylab="y", asp=1)
```



```r
count <- genocount(X=crypho)
count$n
```

```
## [1] 92
```

A total of 92 genotypes is present in the dataset. We then compute the matrix of weights:

```r
mat <- genoweight(X=crypho,genotyp=count$vec)
class(mat)
```

```
## [1] "dist"
```

```r
dim(as.matrix(mat))
```

```
## [1] 276 276
```

The output of `genoweight` is an object of class `dist`. It corresponds to a matrix of 276 × 276, the number of individuals in the dataset. The weighted variogram is computed as follows:

```r
d <- distlag(dist=crypho$coord, dmin=0,distance.lag=50)
d
```

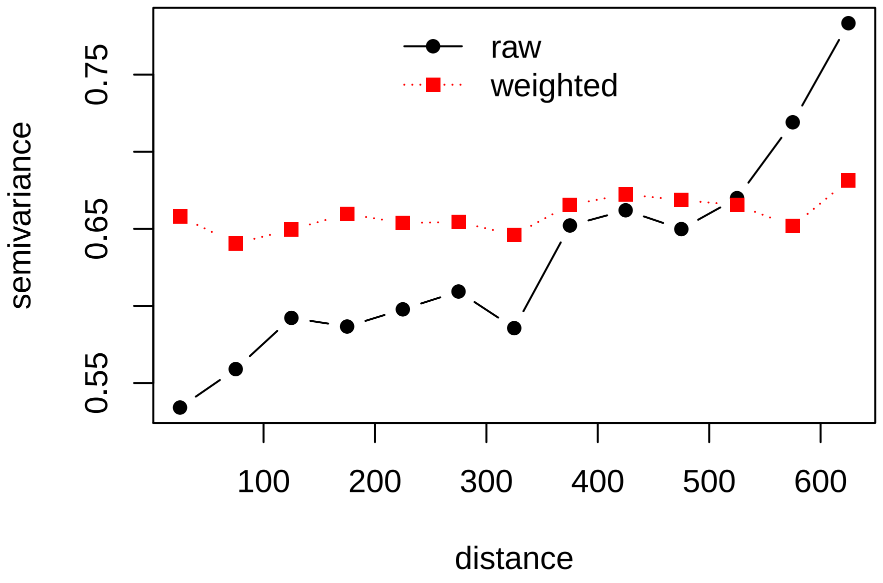
```
##  [1]  25  75 125 175 225 275 325 375 425 475 525 575 625
```

```
wva <- varioWeight(X=crypho, weights=mat,  uvec=d)
```

`varioWeight` returns both the semivariance with and without weighting for recurrent genetypes. The regular semivariance is returned under the slot `$gamma` (`wa$svario$gamma` in the example). It is strictly similar to the values returned by the function `svariog`. semivariance for weighted dataset is given in the slot `$v` (`wa$svario$v` in the example above).

We can plot the variograms:

```
#plot the weighted variogram
plot(wva$svario$u, wva$svario$gamma, col="black", type="b",
     ylim=range(c(wva$svario$gamma,wva$svario$v)), pch=16,
     xlab="distance", ylab="semivariance")
#add the variogram for raw data
points(wva$svario$u, wva$svario$v, col="red", type="b", pch=15,
       lty="dotted")

legend("top", legend=c("raw", "weighted"), col=c("black", "red"),
       pch=c(16,15), lty=c("solid", "dotted"), bty="n")
```

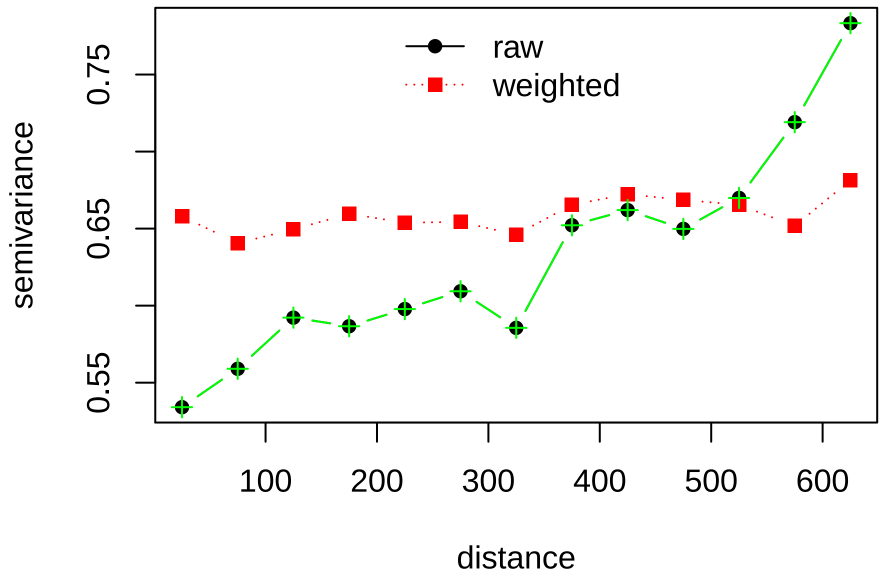


We can also use the function `svariog`:

```
va <- svariog(X=crypho, uvec=d, plot=FALSE)
```

The resulting variogram is similar to the object returned by `varioWeight`:

```
#plot the weighted variogram
plot(wva$svario$u, wva$svario$gamma, col="black", type="b",
     ylim=range(c(wva$svario$gamma,wva$svario$v)), pch=16,
     xlab="distance", ylab="semivariance")
#add the variogram for raw data
points(wva$svario$u, wva$svario$v, col="red", type="b", pch=15,
       lty="dotted")

legend("top", legend=c("raw", "weighted"), col=c("black", "red"),
       pch=c(16,15), lty=c("solid", "dotted"), bty="n")

points(va$svario$u, va$svario$v, col="green", type="b", pch=3, bg="green")
```



From this figure it is obvious that weighting for reccurrent genotypes has a strong impact on the resulting variogram. The weighted variogram shows a very low amount of struture and possibly no SGS at all.

5.2.3.2 Statistical envelopes for weighted variograms: the argument `weights` of `svariog`

The example below shows how to compute the weighted variogram for the dataset `crypho` and how to perform randomization of the weighted semivariance. Users must provide the weighting matrix as the argument `weights` in addition to an object of the class `ggene`.

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

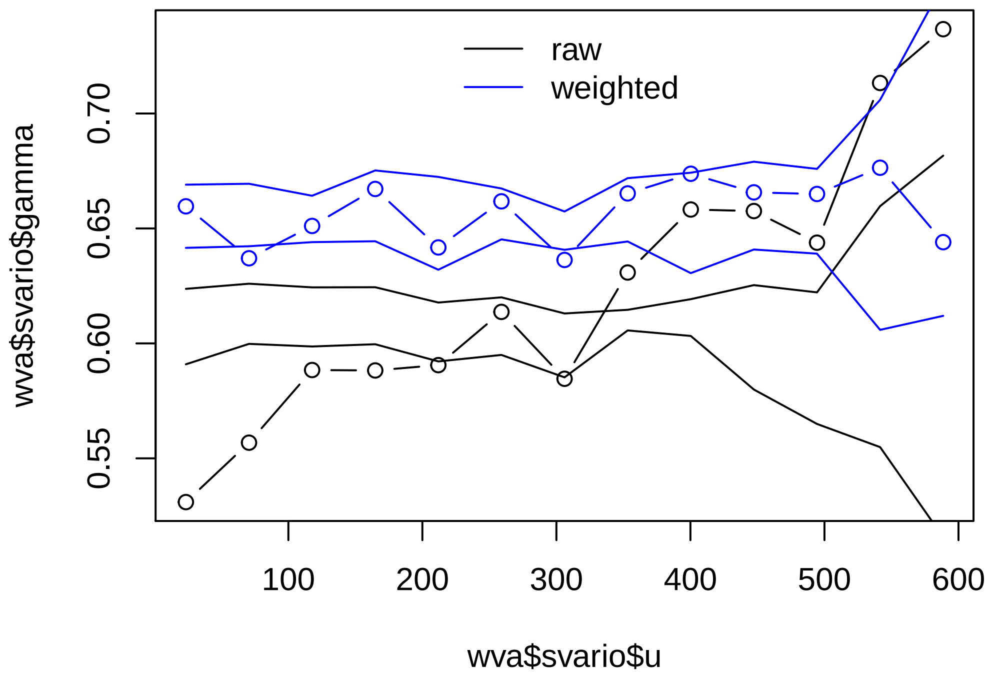
```
#performs the randomizations on raw variogram
va <- svariog(X=crypho, plot=FALSE)
env <- randsvariog(var=va, X=crypho, nsim=9, bounds=NULL,
                   save.sim=FALSE)
```

```
## .........
## done
```

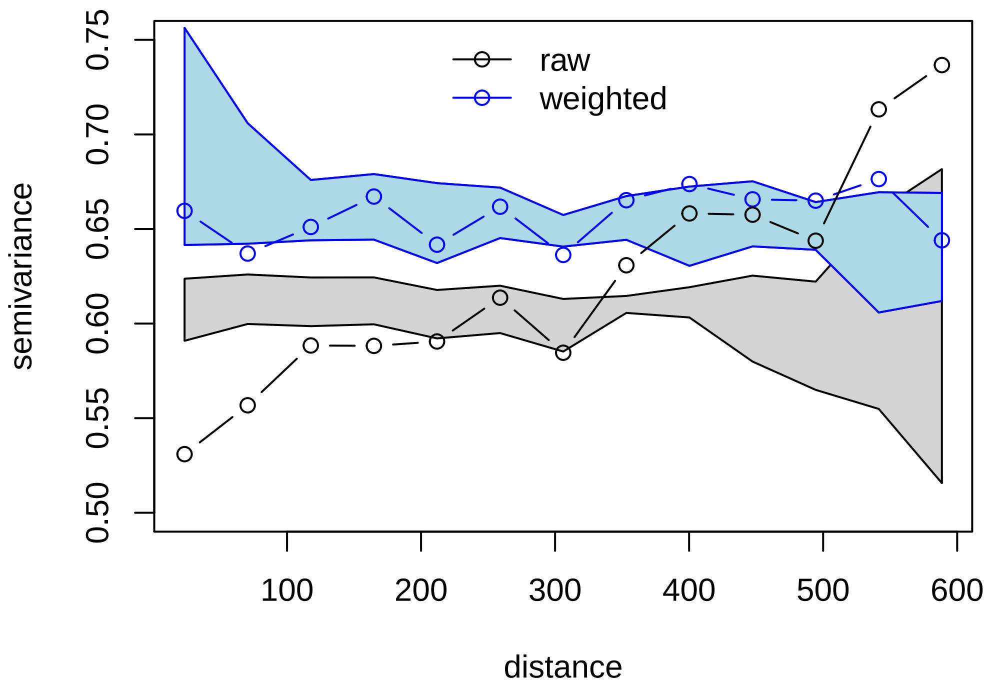
```
#compute the weighted variogram
wva <- varioWeight(X=crypho, weights=mat)

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

```
## .........
## done
```

```
#draw the variogram (raw and weighted) and their envelopes
plot(wva$svario$u, wva$svario$gamma, type="b", col="black")
points(env$svario$u, env$env[,1], type="l", col="black")
points(env$svario$u, env$env[,2], type="l", col="black")

points(wva$svario$u, wva$svario$v, col="blue", type="b",
       ylim=range(c(wva$svario$gamma,wva$svario$v)))
points(env2$svario$u, env2$env[,1], type="l", col="blue")
points(env2$svario$u, env2$env[,2], type="l", col="blue")
legend("top", legend=c("raw", "weighted"), col=c("black", "blue"),
       lty="solid", bty="n")
```



```
# same but another style...
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=c(0.5,0.75))
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")
legend("top", legend=c("raw", "weighted"), col=c("black", "blue"),
       lty="solid", pch=1, bty="n")
```

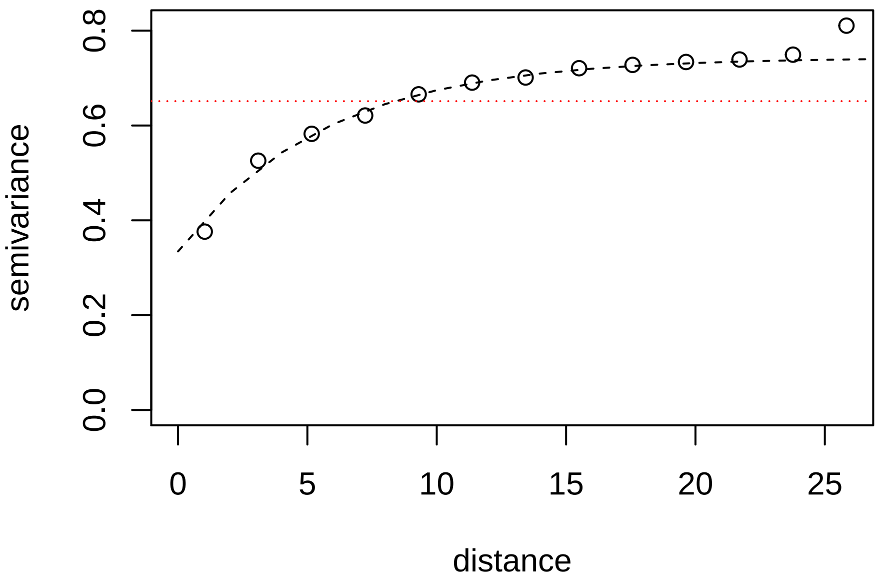


We can conclude that repeated genotypes and the way they are spatially distributed is the main source of SGS in this dataset (Dutech et al. 2008).

5.3 Fitting models to the empirical variogram

There is a wealth of literature dedicated to model fitting in geostatistics (McBratney and Webster 1986; Journel and Huijbregts 1978; Isaaks and Srivastava 1989; Webster and Oliver 1990; Goovaerts 1997). Basically, one of the authorized functions is fitted to the empirical variogram. The model parameters provide interesting information on the spatial structure at hand. They are also used for interpolation by kriging algorithm, a point that is beyond the scope of the present document. Wagner et al. (2005) showed that under certain assumptions, fitting an exponential model to the empirical variogram provides a summary of the SGS. In `ggene`, model fitting is realized by the function `fitsvariog`. It returns the usual variogram parameters (partial sill, nugget variance, range, practical range, see *e.g.* Goovaerts (1997)) as well as the indices developped specifically with the aim to characterize the SGS (F_N, b_f ans S_p, (Vekemans and Hardy 2004; H. Wagner et al. 2005)).

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



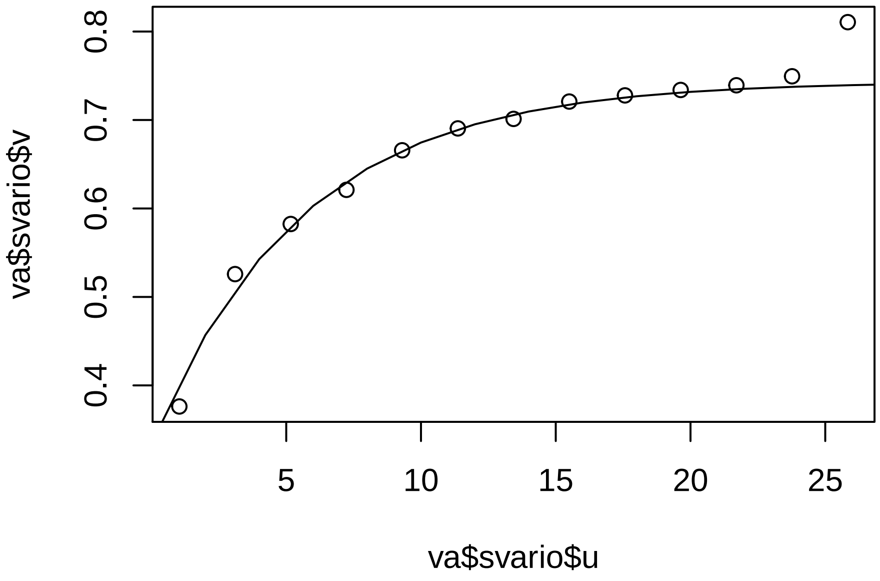
The red dotted line indicates the conventional estimation of the variance (gene diversity) $\hat{H}$. It is returned by `svariog`. The model parameters can be accessed by:

```
fit$param
```

```
##           c1    nugget    range pract.range      sill      Hhat         FN
## 1 0.4089432 0.3344568 5.614505    16.84351 0.7433999 0.651282 0.5500985
##            bf         Sp
## 1 -0.1781101 0.3958868
```

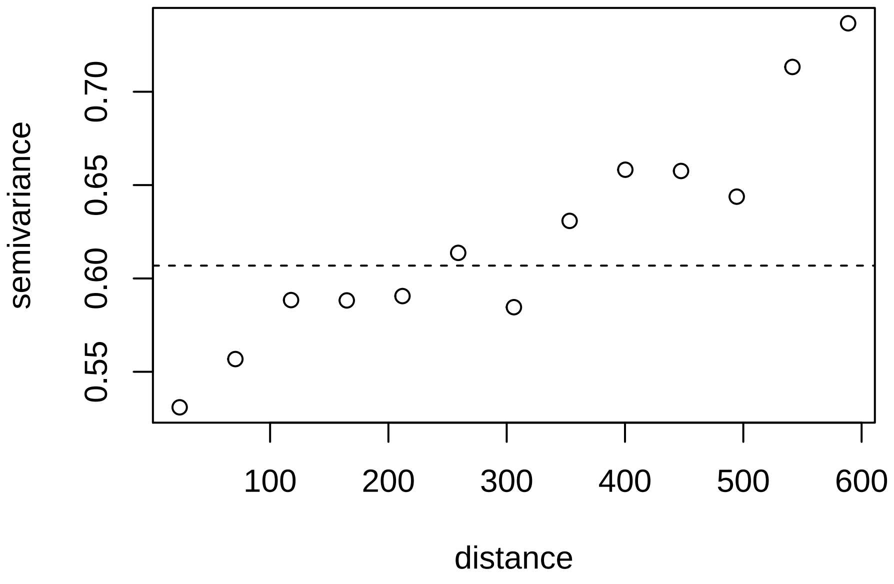
The object `fit` can be directly used with `lines` to plot the model:

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

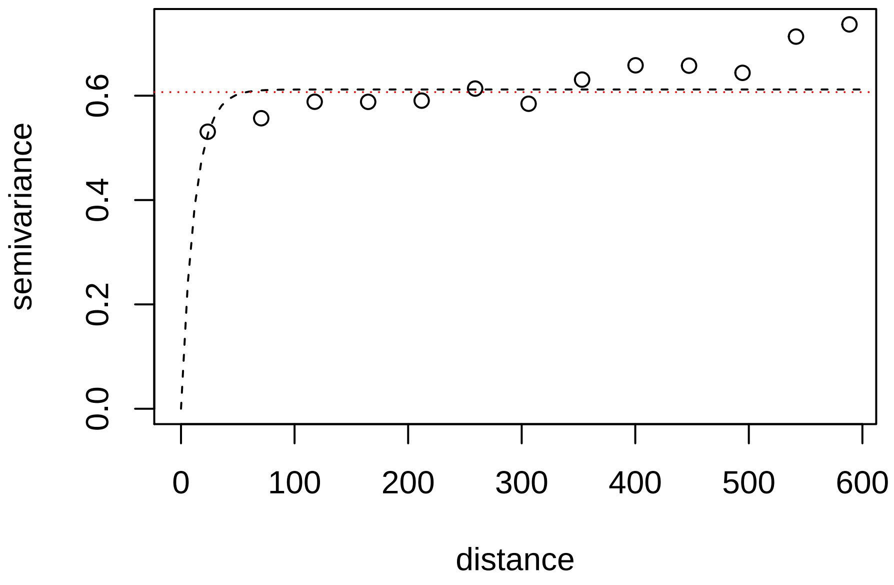


The example above is based an a simulated dataset and the fit is very good. Let's now examine an empirical datset.

```r
data(crypho)
va <- svariog(X=crypho, plot=TRUE, messages=FALSE)
```



```r
fit <- fitsvariog(vario=va, ini.cov.pars=c(0.05,4.5), nugget=0.5,
                  max.dist=600, plot = TRUE)
```

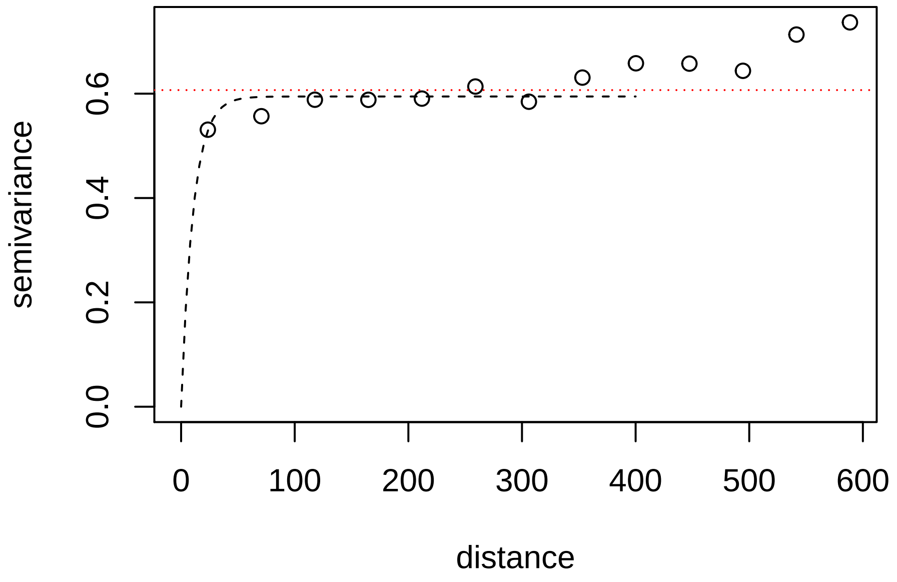


```r
fit$param
```

```
##          c1 nugget  range pract.range      sill      Hhat FN          bf
## 1 0.6119464      0 11.986    35.95799 0.6119464 0.6068645  1 -0.0834307
##     Sp
## 1 Inf
```

The practical range is larger than 235 km ! We get another value if we reduce the maximum distance over which the fit is done.

```
fit <- fitsvariog(vario=va, ini.cov.pars=c(0.05,4.5), nugget=0.5,
                  max.dist=400, plot = TRUE)
```



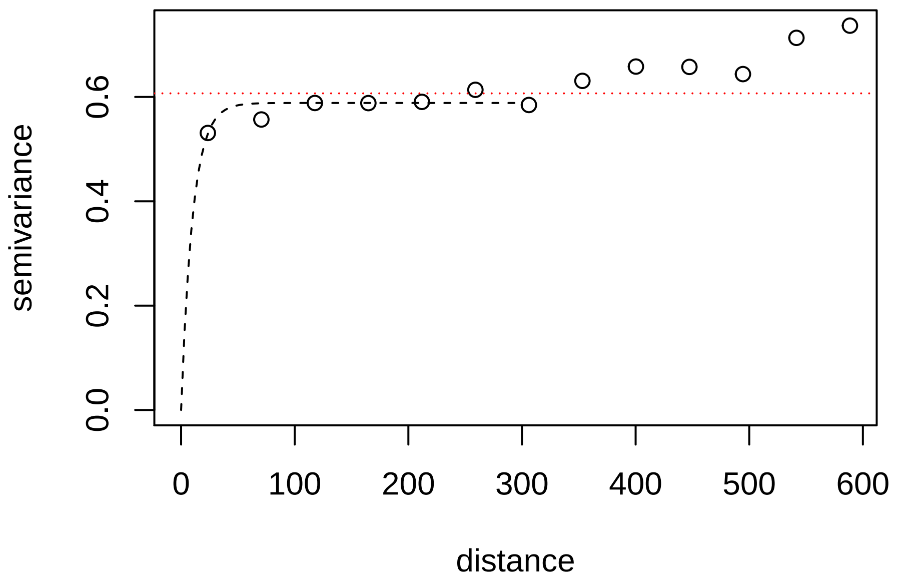
```
fit$param
```

```
##          c1 nugget    range pract.range      sill      Hhat FN           bf
## 1 0.5945829      0 10.68576    32.05728 0.5945829 0.6068645  1 -0.09358248
##     Sp
## 1 Inf
```

The practical range is estimated as 393 m.

Let’s reduce again the maximum distance over which the fit is done.

```
fit <- fitsvariog(vario=va, ini.cov.pars=c(0.05,4.5), nugget=0.5,
                  max.dist=300, plot = TRUE)
```



```
fit$param
```

```
##          c1 nugget    range pract.range      sill      Hhat FN          bf
## 1 0.5884712      0 10.22987     30.6896 0.5884712 0.6068645  1 -0.09775298
##    Sp
## 1 Inf
```

We get another value of ca. 330. This example illustrates how choosing the distance range over which the model is fitted can affect the model parameters. Why is this aspect so important in the case of *Cryphonectria parasitica*? The empirical variogram shows the presence of a first plateau for distances ranging from ca. 100 m up to ca. 300 m but the semivariance increases again for larger distances. This means that SGS spotted at distance <300 m is nested into long-range pattern. Such pattern could be seen for example if local demographical processes create a local SGS itself superimposed on a longer range isolation by distance structure. Caution is needed as it is very difficult to infer a pattern from its summary by a structure function such as the variograms or the correlograms. Regarding fitting models, a good practice consists in fitting the model for the first distance classes and restrain the computation to values below the first half or third of the maximum distance. This is a rule of thumb commonly used in geostatistics (Isaaks and Srivastava 1989; Journel and Huijbregts 1978) that ensures that the fit is good at least for the most local structures, an important thing regarding kriging.

6 Dealing with anisotropy

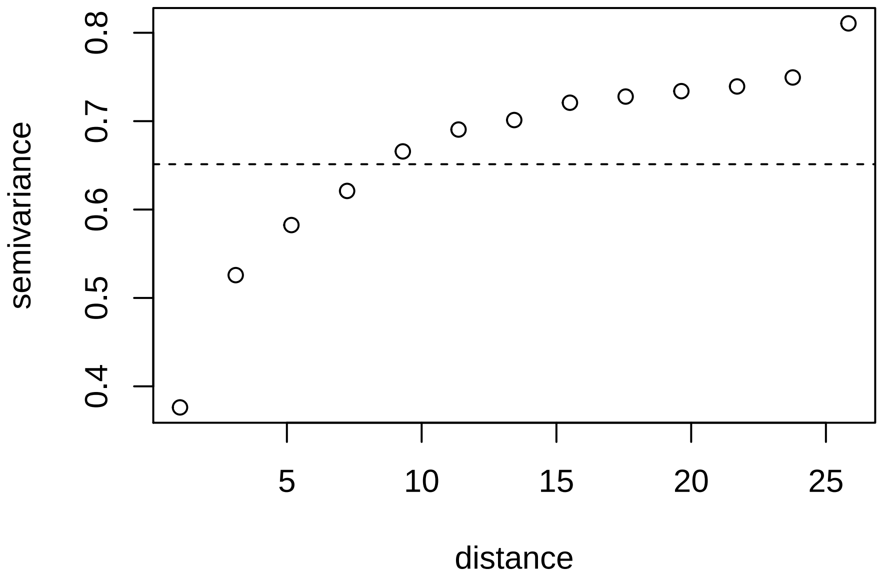
Anisotropy is the property of being directionally dependent. It is opposed to isotropy. Anisotropy occurs if the spatial pattern differs, when measured along different axes, either in extent or intensity. **Geometric anisotropy** corresponds to situations where variograms have the same sill in all directions through their ranges are different. **Zonal (or stratified anisotropy)** occurs when sills are different according to the direction considered. `ggene` allows searching and charaterizing anisotropies of genetic variation either by computing the so-called directional variograms or by drawing variogram maps (Isaaks and Srivastava 1989).

6.1 Directional variograms

`svariog` computes omnidirectional variograms by default. The arguments `direction`, `tolerance` and `unit.angle` allow to define the main direction, the tolerance and the unit of measure for the angle. These arguments, along others, are passed to the function `variog` from package `geoR`. Users are referred to the documentation associated to this function for details on computation options and default values.

Computation of the directional variogram

```r
data(aniso)
va <- svariog(X=aniso, plot=TRUE)
```

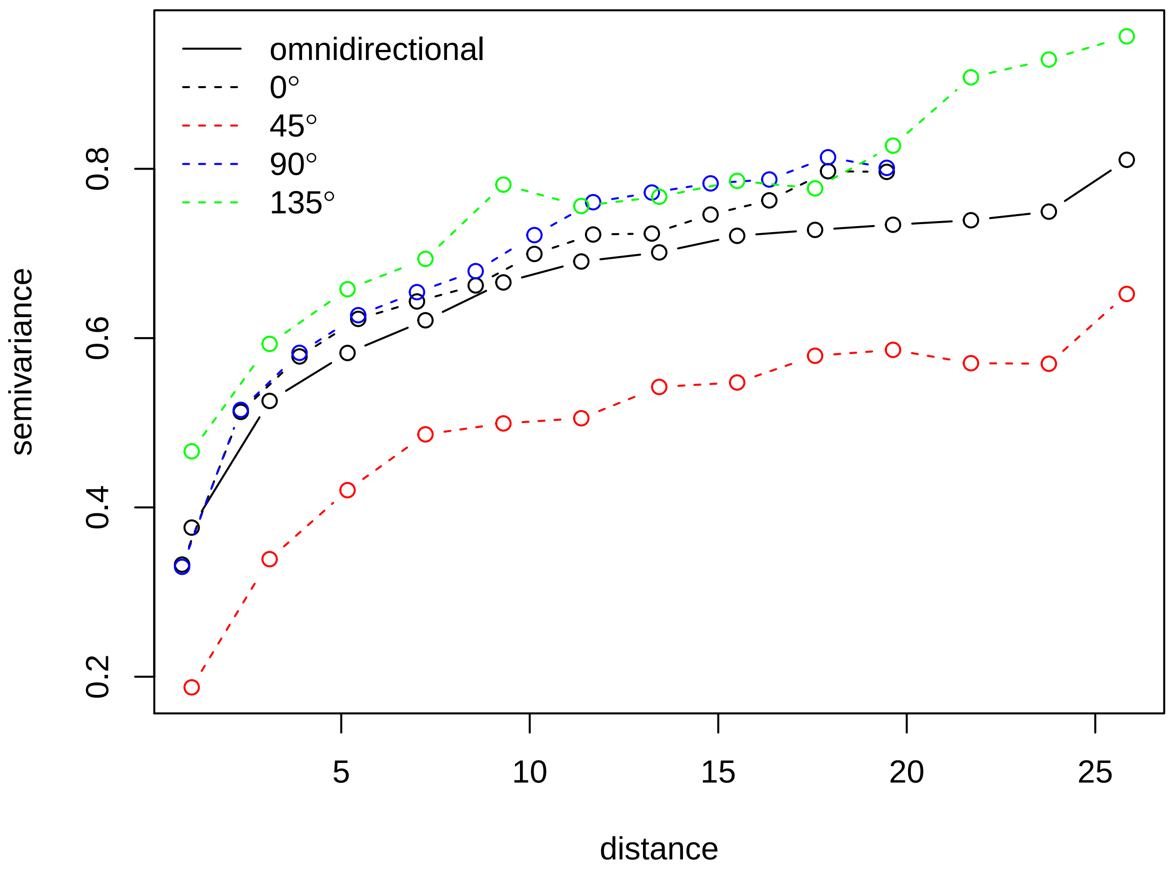


The omni-directional variogram reveals a clear spatial genetic structure. Does this SGS differ according to the main directional axes considered? We consider 4 directions : 0, 45, 90 and 135 degrees. The same tolerance of 22.5° is considered in each case.

```r
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")
```

The variograms can be plotted on the same graph:

```
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")
```



From the plot, it can be seen that the semivariance is markedly lower in the direction 45°. The shape of the variogram is very similar but the magnitude differs. The example illustrates a case of zonal anisotropy.

6.2 Variogram maps

Searching for directional anisotropies can also be achieved using a display called the variogram map *aka* variogram surface (Isaaks and Srivastava 1989, 149). Pairs of individuals are constituted by gathering individuals whose separation in the x direction is $h_x \pm \delta_x$ and whose separation in the y direction is $h_y \pm \delta_y$. The grid considered has cells of width δ_x and a overall length called cutoff. In `ggene` the function `svarmap` computed variogram maps on the basis of a `ggene` object, a cutoff value (*i.e.* the maximum distance to be considered) and the width *i.e.* δ_x (only square cells are implemented which means that $\delta_x = \delta_y$)(figure 4 page 46).

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

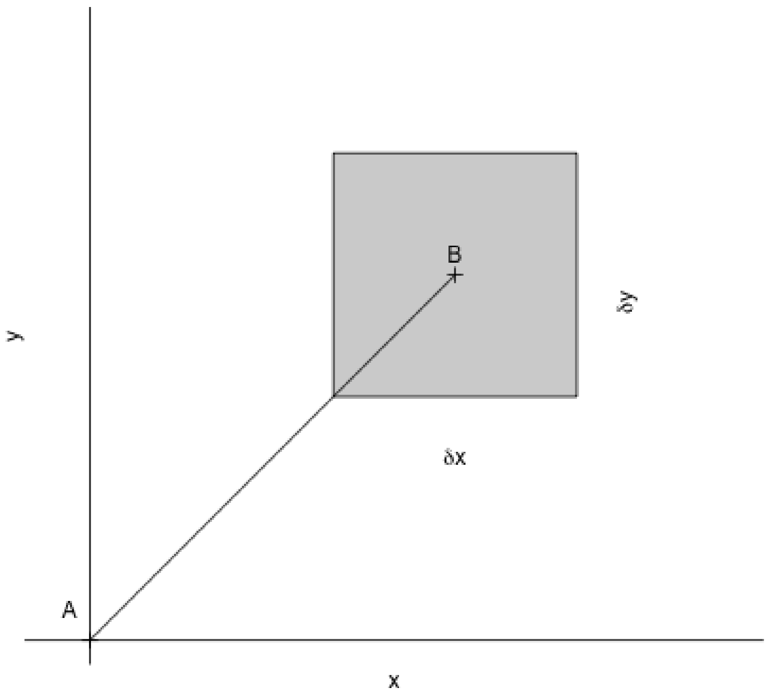
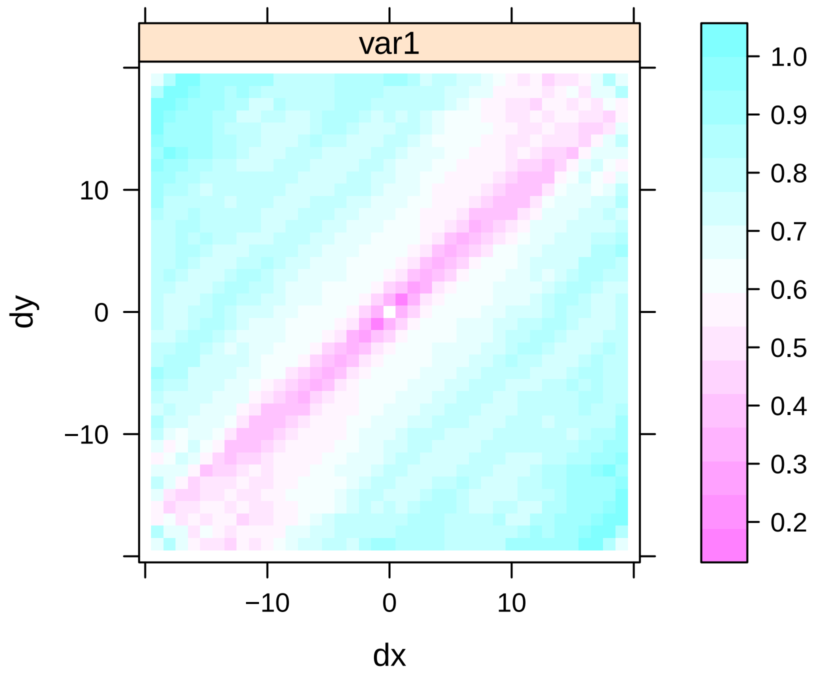
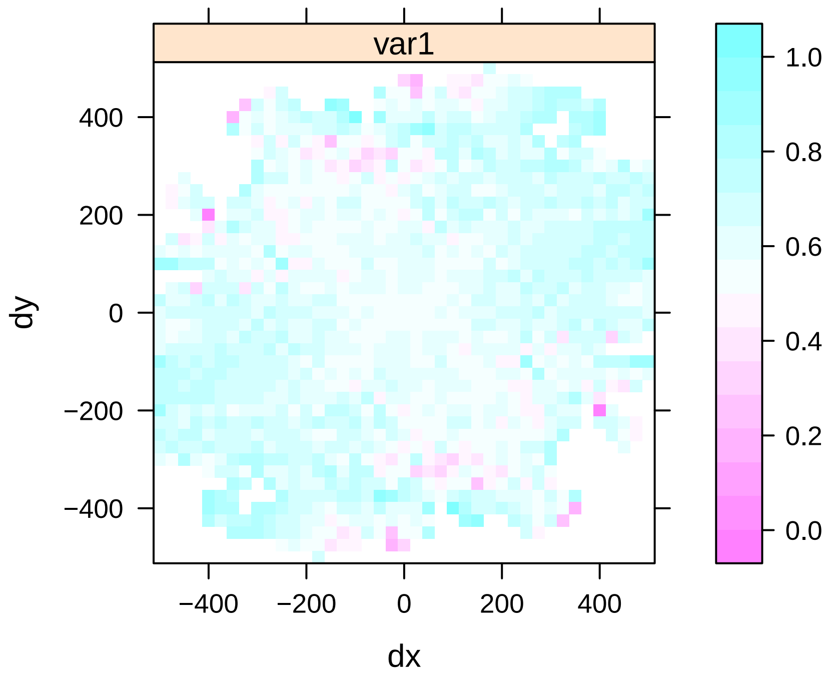


Figure 4: Grouping points to form a variogram map. Individuals at B and all other individuals falling within the shaded area will be paired with the individual at A. The semivariance of all such pairs are averaged and plotted in a raster map.



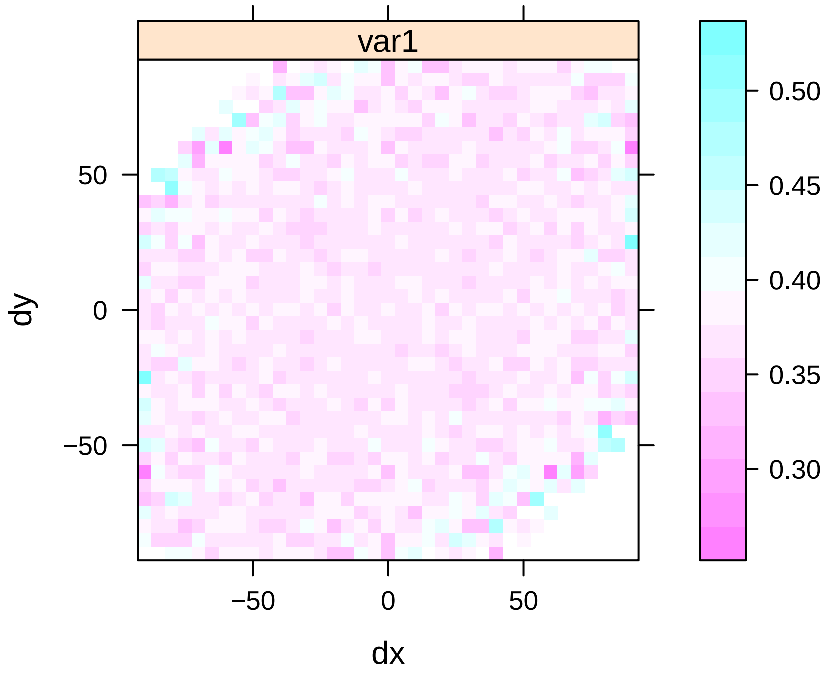
As with the directional variograms, the variogram map reveals the presence of a direction for which the semivariance (here gene diversity) is lower (pink areas on the map). This direction is 45°. Note that the results of the analysis is particularly clear because the dataset `aniso` was simulated and as such exhibits a very strong pattern. Let's examine the variogram map for the empirical dataset `crypho`.

```
map <- svarmap(X=crypho,cutoff=500, width=25)
plot(map)
```



There is no clear directional pattern although pixels at the centre of the plot show lower semivariance values. This simply conveys the isotropic spatial structure identifed above. What happens when no struture is present *at all*? We can return to the `larix1350` dataset (for which no pattern was found) and compute the variogram map.

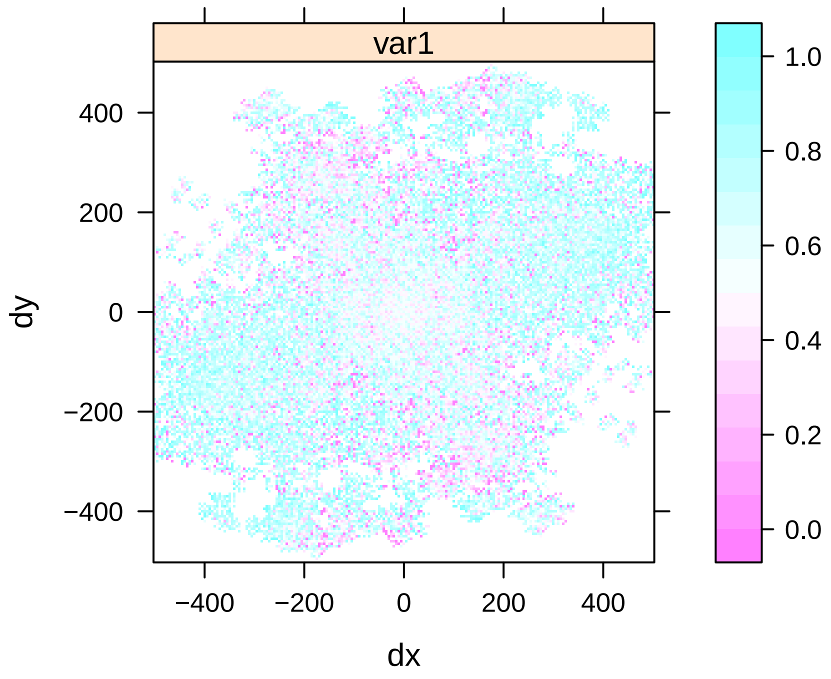
```
data(larix1350)
map <- svarmap(X=larix1350,cutoff=90, width=5) ; plot(map)
```



We here see that there is no clear pattern part from lower values at the margin of the plot. These are probably associated to estimates based on few data pairs, which is a common feature of variograms. We will deal with this question below.

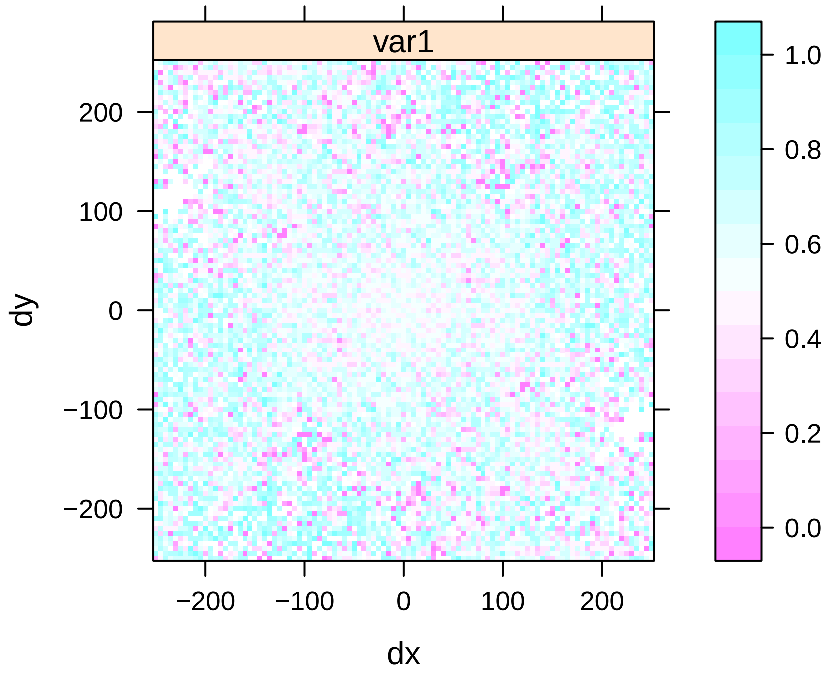
The value of the width parameter affects the grain of the outputs. Larger values smooth up the semivariance which can obscure our perception of the SGS.

```
map <- svarmap(X=crypho,cutoff=500, width=5)
plot(map)
```



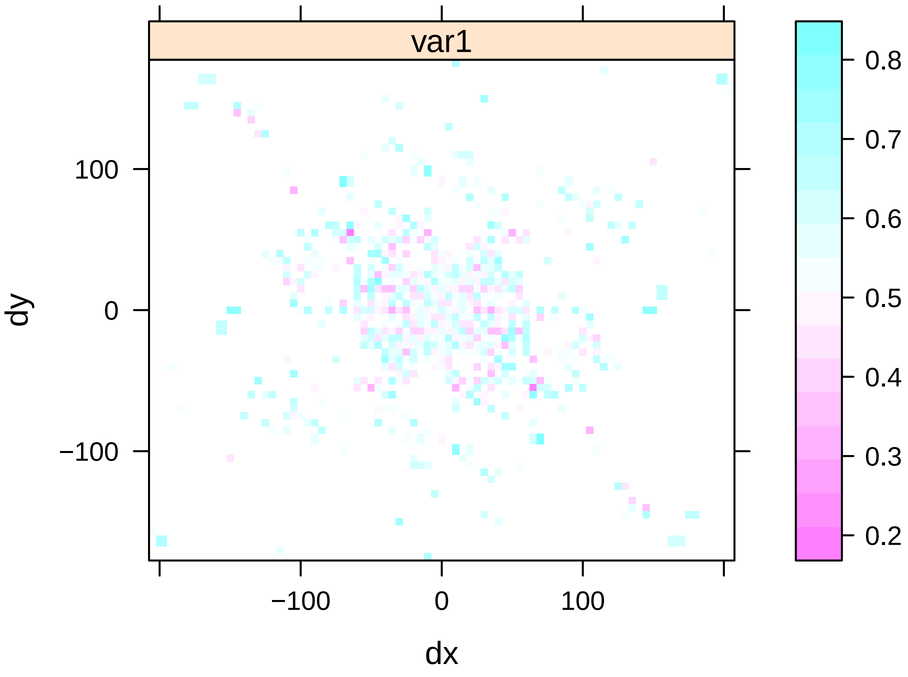
Cutoff value simply limits the extent of the analysis. Larger extents provide a more complete picture but require more computation time.

```
map <- svarmap(X=crypho, cutoff=250, width=5)
plot(map)
```



Whatever the values of `width` and `cutoff` there is no indication of anisotropy. Some cells may contain only few data pairs, this depends on the sampling scheme. Because those cells may contain unreliable semivariance values, they shall be removed. The parameter `threshold` (default value = 5) can be used for that purpose:

```r
plot(map, threshold = 10)
```



Semivariance values derived from a number of data pairs lower than the threshold are not printed on the map.

7 Aknowledgements

We thank C. Dutech and P. Rozenberg who kindly provided the datasets for *Cryphonectria parasitica* and *Larix decidua*.

References

Bellier, E., P. Monestiez, J.P. Durbec, and J.N. Candau. 2007. "Identifying Spatial Relationships at Multiple Scales: Principal Coordinates of Neighbour Matrices (PCNM) and Geostatistical Approaches." *Ecography* 30: 385–99.

Bivand, R.S., E. Pebesma, and V. Gómez-Rubio. 2008. *Applied Spatial Data Analysis with R.* Springer.

Burrough, P.A. 1983. "Problems of Superimposed Effects in Statistical Study of the Spatial Variation in Soil." *Agricultural Water Management* 6: 123–43.

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

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.

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

Hardy, O.J., and X. Vekemans. 1999. "Isolation by Distance in a Continuous Population: Reconciliation Between Spatial Autocorrelation Analysis and Population Genetics Models." *Heredity* 83: 145–54.

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

Jiménez, J.J., T. Decaëns, P. Lavelle, and J.-P. Rossi. 2014. "Dissecting the Multi-Scale Spatial Relationship of Earthworm Assemblages with Soil Environmental Variability." *BMC Ecology* 14: 26.

Jombart, T. 2008. "Adegenet: A R Package for the Multivariate Analysis of Genetic Markers." *Bioinformatics* 24: 1403–5.

Journel, A.G., and C.J. Huijbregts. 1978. *Mining Geostatistics*. Academic Press.

Le Corre, V., G. Roussel, A. Zanetto, and A. Kremer. 1998. "Geographical Structure of Gene Diversity in *Quercus Petraea* (Matt.) Liebl. III. Patterns of Variation Identified by Geostatistical Analyses." *Heredity* 80: 464–73.

Legendre, P., and L. Legendre. 1998. *Numerical Ecology*. Elsevier.

McBratney, A.B., and R. Webster. 1986. "Choosing Functions for Semi-Variograms of Soil Properties and Fitting Them to Sampling Estimates." *Journal of Soil Science* 37: 617–39.

Nardin, M., B. Musch, Y. Rousselle, V. Guérin, L. Sanchez, J.-P. Rossi, S. Gerber, S. Marin, L. Pâques, and P. Rozenberg. 2015. "Genetic Differentiation of European Larch Along an Altitudinal Gradient in the French Alps." *Annals of Forest Science* 72: 517–27.

Oliver, M.A. 2010. *Geostatistical Applications for Precision Agriculture*. Springer.

Rossi, J.-P., M. Nardin, M. Godefroid, M. Ruiz-Diaz, A.-S. Sergent, A. Martinez-Meier, L. Pâques, and P. Rozenberg. 2014. "Dissecting the Space-Time Structure of Tree-Ring Datasets Using the Partial Triadic Analysis." *PLoS ONE* 9: e108332.

Vekemans, X., and O.J. Hardy. 2004. "New Insights from Fine-Scale Spatial Genetic Structure Analyses in Plant Populations." *Molecular Ecology* 13: 921–35.

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: 1739–52.

Webster, R., and M.A. Oliver. 1990. *Statistical Methods in Soil and Land Resource Survey*. Oxford University Press.

Werth, S., H. H. Wagner, R Holderegger, Kalwij J.M, and Scheidegger C. 2006. "Effect of Disturbances on the Genetic Diversity of an Old-Forestoold-Fassociated Lichen." *Molecular Ecology* 15: 911–21.