Datasets provided in 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 Lists of datasets

1.1 ggene objects

The package comes with 8 datasets used to examplify the functions and show how the results can be interpreted. These datasets can be accessed using the function `data(name of the dataset)`. The table below provides a brief description of each dataset. Typing `?name of the dataset` will open the dedicated help page.

name	description
`aniso`	simulated haploid genotypic dataset exhibiting zonal anistropy
`crypho`	haploid microsatellite dataset for the chestnut blight fungus *Cryphonectria parasitica* (Dutech et al. 2008)
`larix1350`	diploid dataset corresponding to a set of 199 European larch trees *Larix decidua* sampled at the altitude 1350 m asl (Nardin et al. 2015)
`larix2300`	diploid dataset corresponding to a set of 175 European larch trees *Larix decidua* sampled at the altitude 2300 m asl (Nardin et al. 2015)
`sim01`	simulated haploid genotypic dataset exhibiting clear spatial structure
`sim02`	simulated haploid genotypic dataset with no spatial structure
`sim03`	a subset of `sim01`
`Wagner`	toy example used in Wagner et al (2005) p. 1750

1.2 Other data types

Several data files are available in the directory `extdata` located in the R installation directory. The path to this directory is indicated by the function `system.file()`:

```
system.file(package="ggene")
```

These files are stored on the `extdata` directory and the path is thus given by typing:

```
system.file("extdata/",package="ggene")
```

name	description
`sim_01.csv`	the file contains the genetic data corresponding to dataset 'sim01' (see above)
`sim_03.gen`	a genepop file with 625 individuals
`xysim_01.csv`	the file contains the x y coordinates of the individuals in the dataset 'sim01' (see above)

The code below shows how the files can be loaded and used in R.

```
library(ggene)
```

```
##
## ggene version 1.0.2 is loaded
```

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

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

```
class(dat.sim)
```

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

Reading genepop files can be achieved using `read.genepop` from the package `adegenet`:

```
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.
```

```
xy <- read.csv(system.file("extdata/xysim_01.csv",package="ggene"),
               header=FALSE)[1:dim(dat$tab)[1],]
data <- gene2geo(X=dat, coord=xy)
```

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

```
class(data)
```

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

```
str(data)
```

```
## List of 5
##  $ tab     : num [1:625, 1:502] 1 0 0 1 1 0 0 0 0 0 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:625] "1" "2" "3" "4" ...
##   .. ..$ : chr [1:502] "l1.024" "l1.038" "l1.039" "l1.018" ...
##  $ 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] "l1" "l2" "l3" "l4" ...
##  $ locnames: Factor w/ 20 levels "l1","l2","l3",..: 1 1 1 1 1 1 1 1 1 1 ...
##  - attr(*, "class")= chr [1:2] "list" "ggene"
```

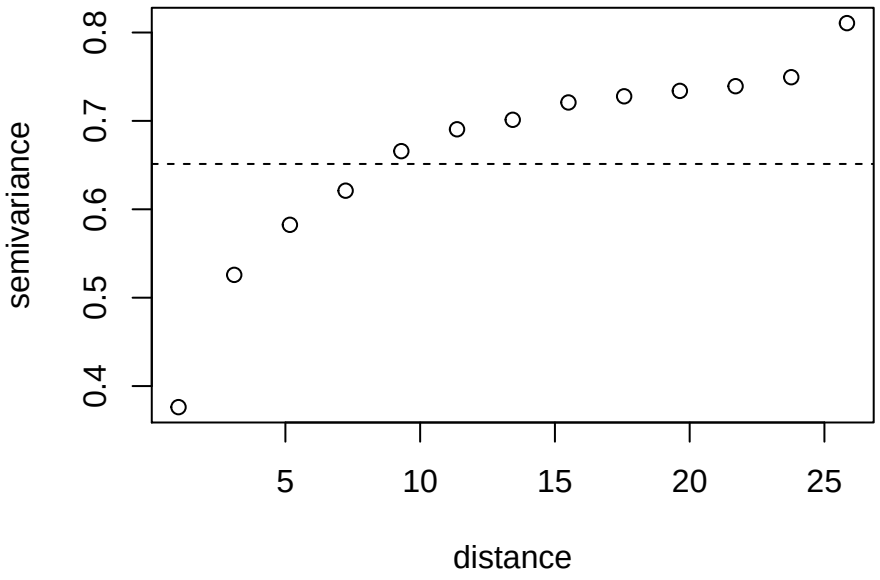
2 Information on each dataset

2.1 aniso

`aniso` is a simulated dataset illustrating how variogram can be used to explore spatial anisotropies of a microsatellite dataset. `aniso` is provided as a `ggene` object. Anistropy can be explored using directional variograms (function `svariog`) or variogram maps (function `varmap`).

Directional variograms are variograms computed for a given set of direction and tolerance (Goovaerts 1997). `svariog` allows a straightforward analysis of anisotropy. First, the omnidirectional variogram is computed using `svariog`. If the argument `plot` is set to TRUE, the function plots the variogram.

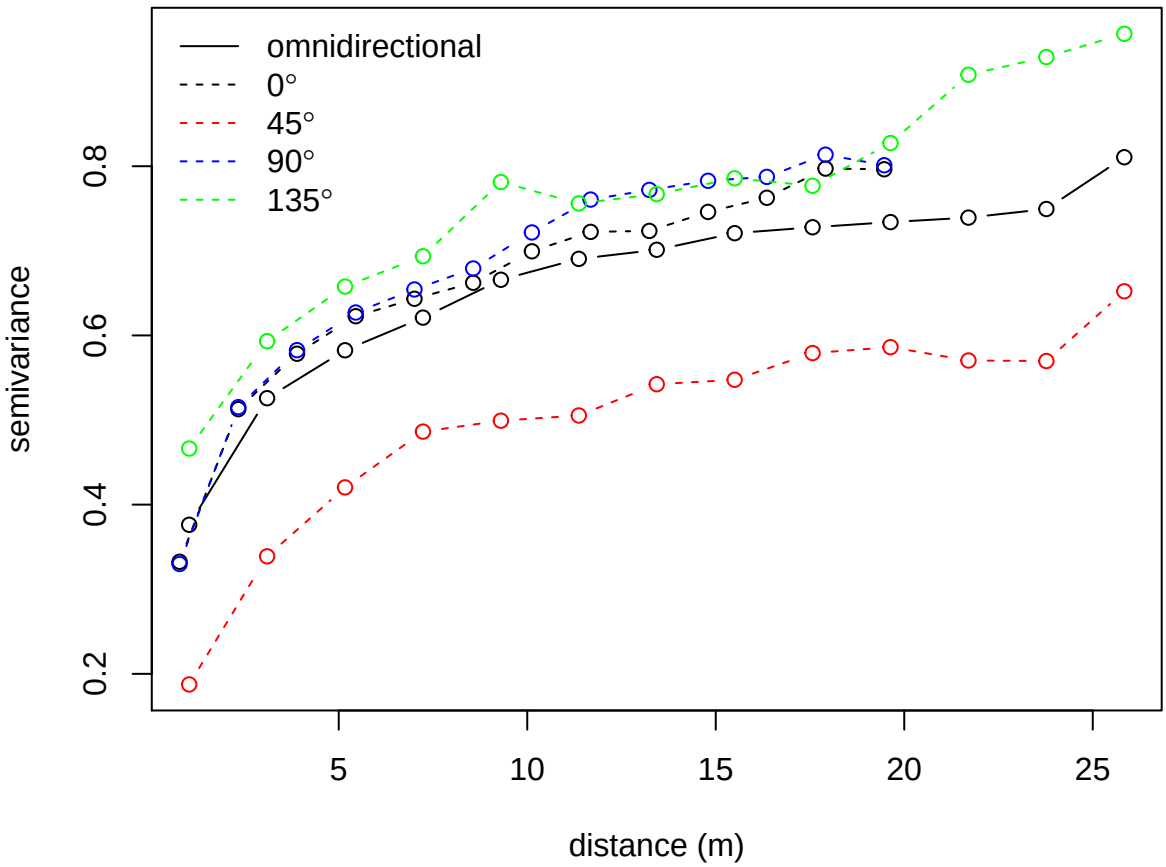
```
library(ggene)
data(aniso)
va <- svariog(X=aniso, plot=TRUE)
```



Each directional variogram corresponds to a direction and a tolerance. We here use 4 directions ranging from 0 to 135 degrees, with a tolerance of 22.5 degrees.

```
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")
```

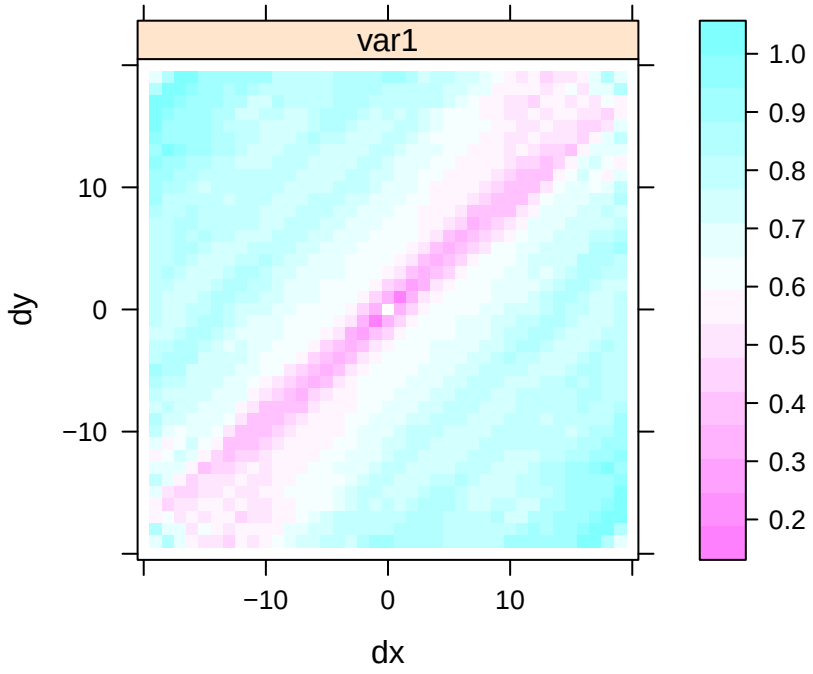
```
plot(va$svario$u, va$svario$v, type="b",
ylim=range(c(va$svario$v, d0_225$svario$v, d45_225$svario$v,
d90_225$svario$v,  d135_225$svario$v)),xlab="distance (m)",
ylab="semivariance")
points(d0_225$svario$u, d0_225$svario$v, type="b", lty=2)
points(d45_225$svario$u, d45_225$svario$v, type="b", col="red", 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")
```



It can be seen that the directional variogram for the direction of 45° exhibits a clear departure from the omnidirectional variogram.

Anisotropy can be assessed using another approach, the variogram map or 'variogram surface' (Isaaks & Srivastava (1989) p. 149). The parameter `cutoff` indicates the maximum lag distance to be considered. `width` is the lag distance increment.

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

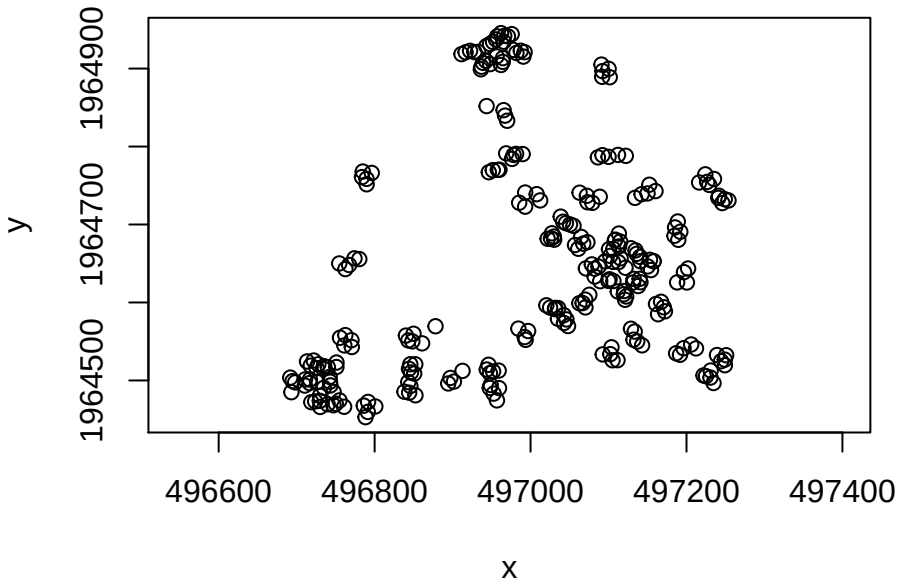
2.2 `crypho`

`crypho` consists of a set of 10 locus for 276 individuals of the chestnut blight fungus *Cryphonectria parasitica* (CBF). This organism is haploid and a certain level of clonality is observed (Dutech et al. 2008). This data set is therefore an interesting case study if we are to explore how repeated genetypes affect the observed genetic variation. Following the proposition of Wagner et al (2005), we will use a weighting scheme to account for recurrent genotypes.

```
data(crypho)
```

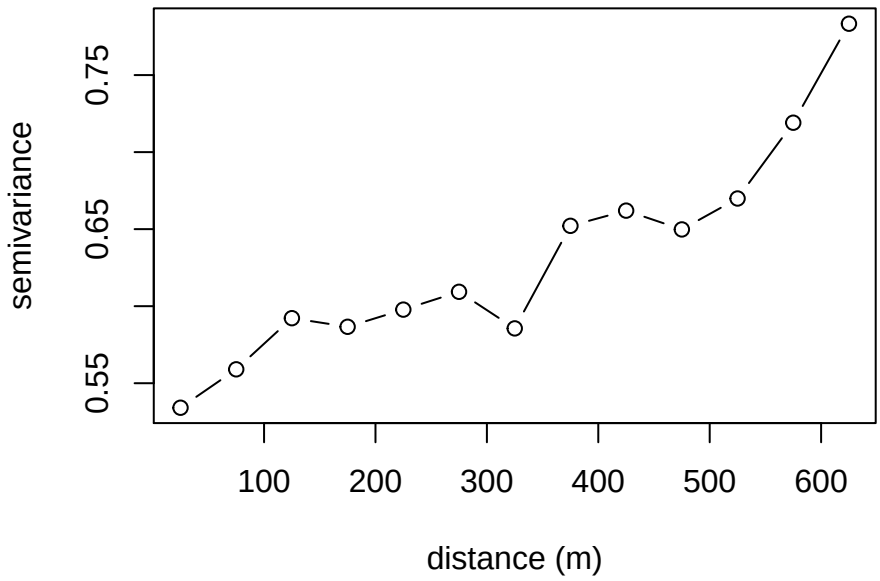
The sampling scheme can be examined prior to analyses. The coordinates of the individuals are stored in the `$coord` slot of the `ggene` object.

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



The variogram can be computed using the function `svariog`:

```
d <- distlag(dist=crypho$coord, dmin=0,distance.lag=50)
va <- svariog(X=crypho, uvec=d, plot=FALSE)
#plot raw variogram
plot(va$svario$u, va$svario$v, col="black", type="b",
      xlab="distance (m)", ylab="semivariance")
```



Computing the weighted variogram requires the matrix of weights. It is produced by 2 functions, `genocount` and `genoweight`.

- `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`.

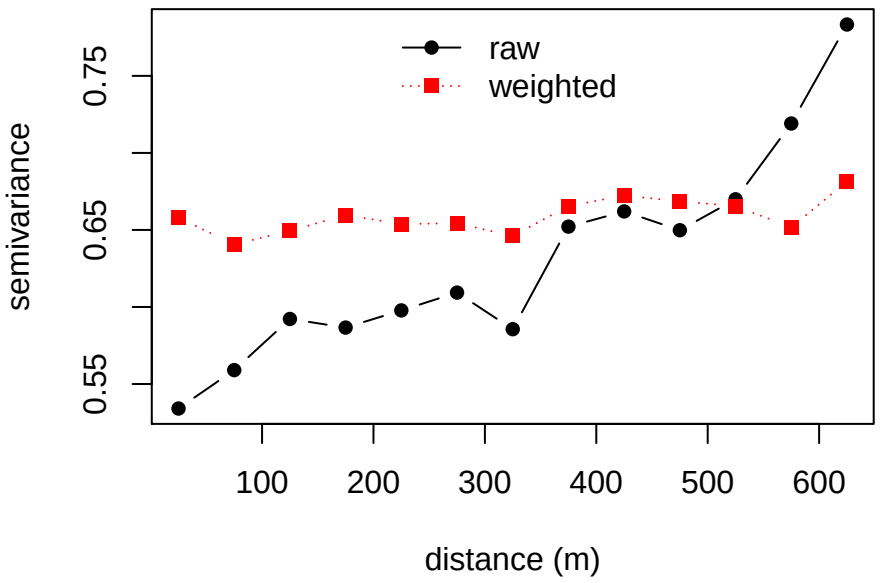
```
# compute matrix of weights
count <- genocount(X=crypho)
mat <- genoweight(X=crypho,genotyp=count$vec)
```

The matrix of weights `mat` is then used to compute the weighted variogram using the function `varioWeight`.

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

We plot the variograms:

```
plot(wva$svario$u, wva$svario$gamma, col="black", type="b",
     ylim=range(c(wva$svario$gamma,wva$svario$v)), pch=16,
     xlab="distance (m)", 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")
```



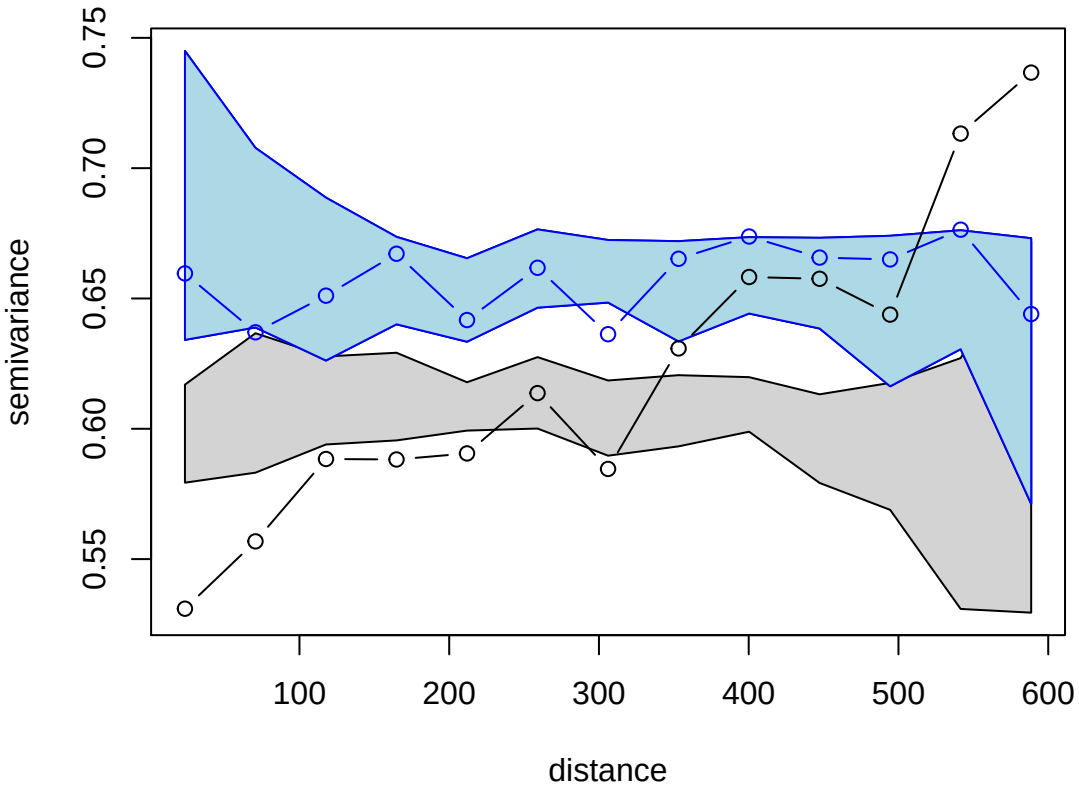
The weighted variogram is flat which revals that accounting for recurrent genotypes strongly decreased the amount of spatial structure.

An interesting question is to explore to which extent the spatial structure has been removed. For that purpose, we can use the function `randsvariog` to compute the statistical envelopes for both raw and weighted variograms.

```
#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)
```

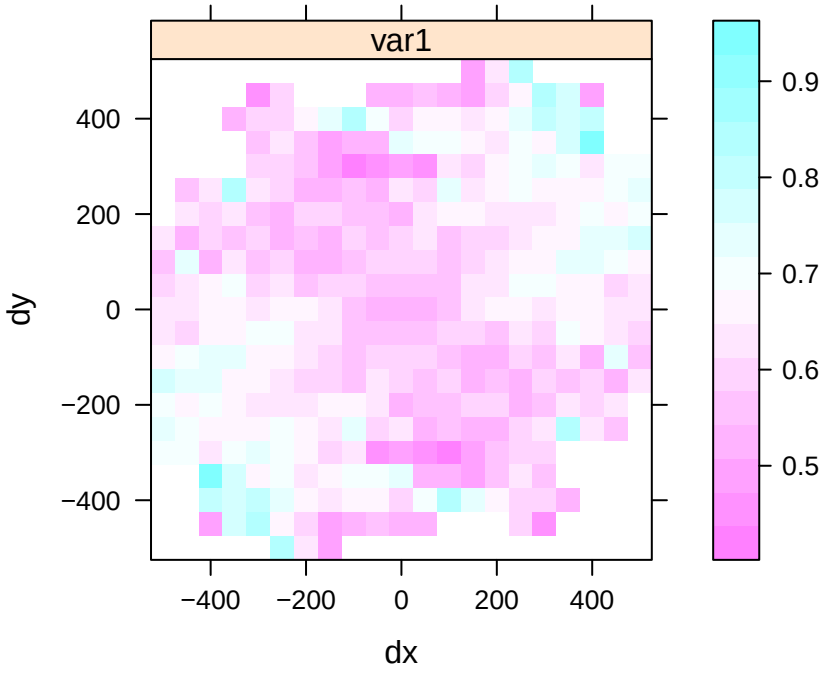
We plot the results:

```
# 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")
```



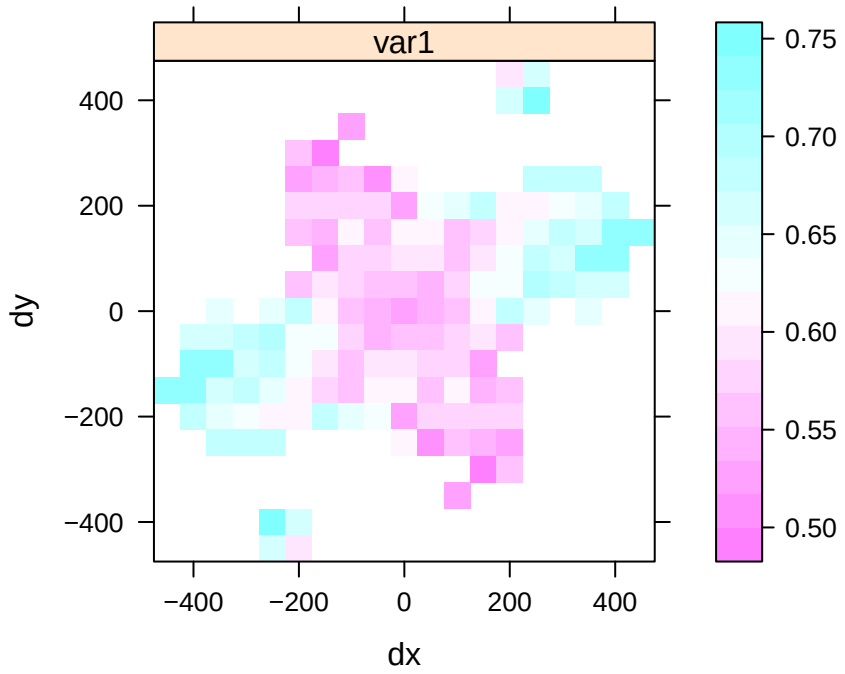
Is the spatial genetic variation of *C. parasitica* isotropic? A simple way to answer the question is to compute the variogram map.

```
 map <- svarmap(X=crypho,cutoff=500, width=50)
plot(map)
```



Because some cells may contain only few data pairs, their associated semivariance estimate might be unriable. These cells can be removed from the map using the parameter `threshold` which default value = 5. Semivariance values derived from a number of data pairs lower than the threshold are not printed on the map.

```
plot(map, threshold=200)
```

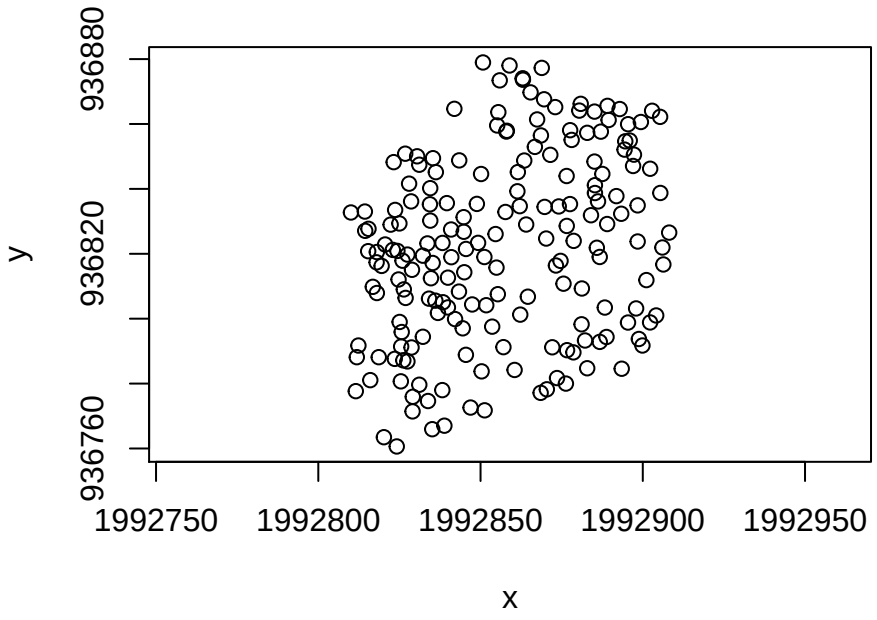


2.3 larix1350

`larix1350` corresponds to the microsatellite dataset for 13 microsatellite locus of 189 individuals of European larch (*Larix decidua*) sampled in an experimental plot at the altitude of 1350 m asl. The survey was undertaken near the village of Villar-Saint-Pancrace (Hautes-Alpes, France) (Nardin et al. 2015).

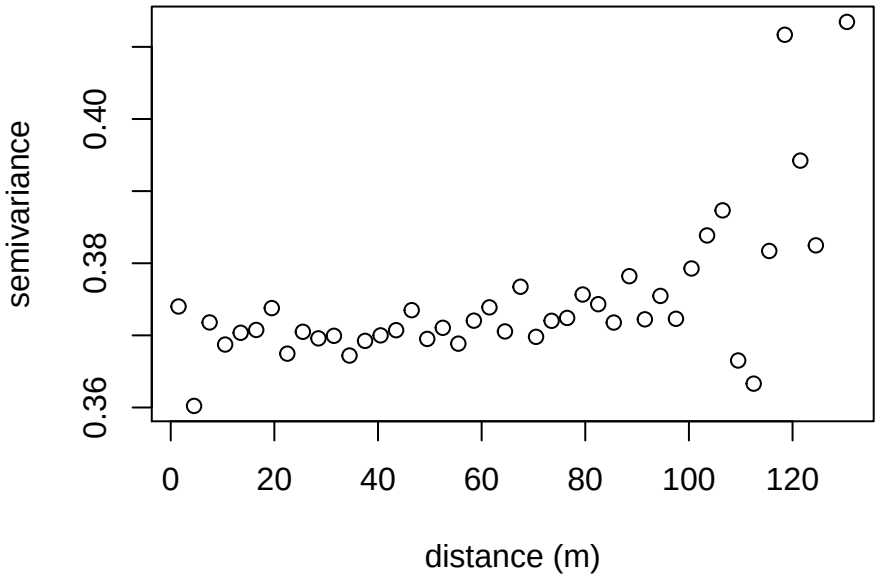
The spatial distribution of the trees can be mapped:

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



The variogram shows very little structuration, if any.

```
# 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, xlab="distance (m)", ylab="semivariance")
```

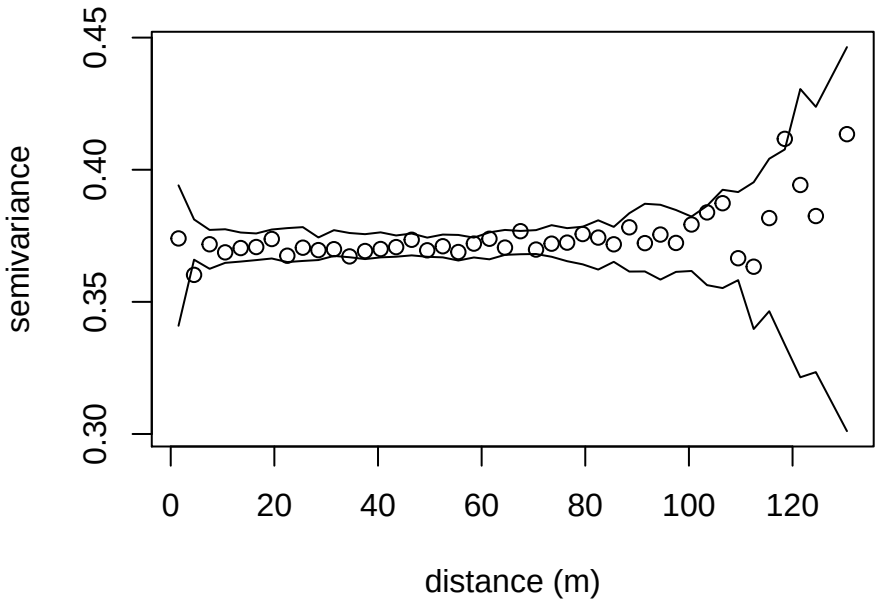


The statistical envelope indicates that most of the semivariance estimates lie between the 0.025 and 0.975 quantiles. Note that the number of randomizations was kept low in this example while regular data analyses should rely on a much larger number.

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

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

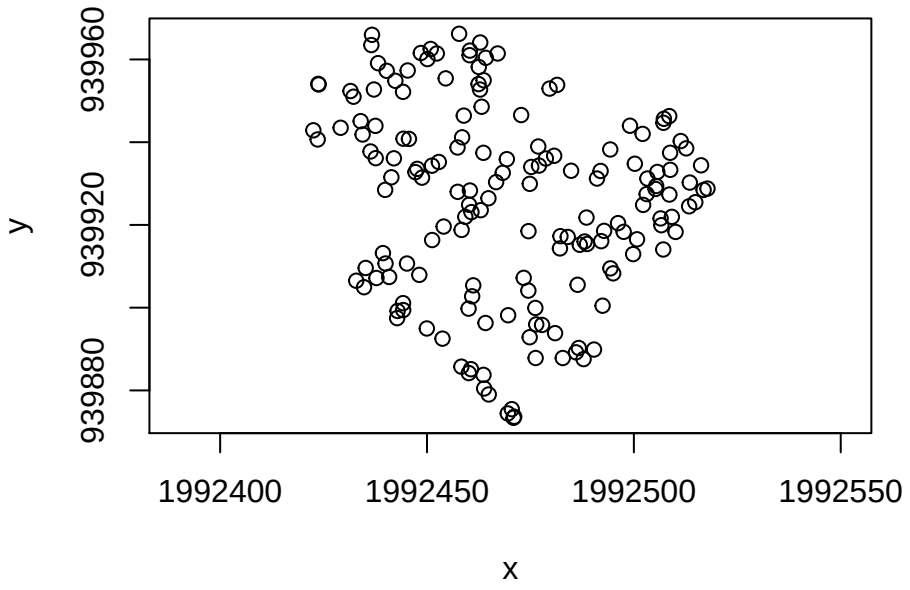
```
# plot results
plot(env$svario$u, env$svario$v, ylim=range(env$env), xlab="distance (m)",
     ylab="semivariance")
points(env$svario$u, env$env[,1], type="l")
points(env$svario$u, env$env[,2], type="l")
```



2.4 larix2300

Another dataset for *L. decidua* collected near the village of Villar-Saint-Pancrace (Hautes-Alpes, France) but this time the sampled population is located at higher alstitude: 2300 m a.s.l (Nardin et al. 2015).

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

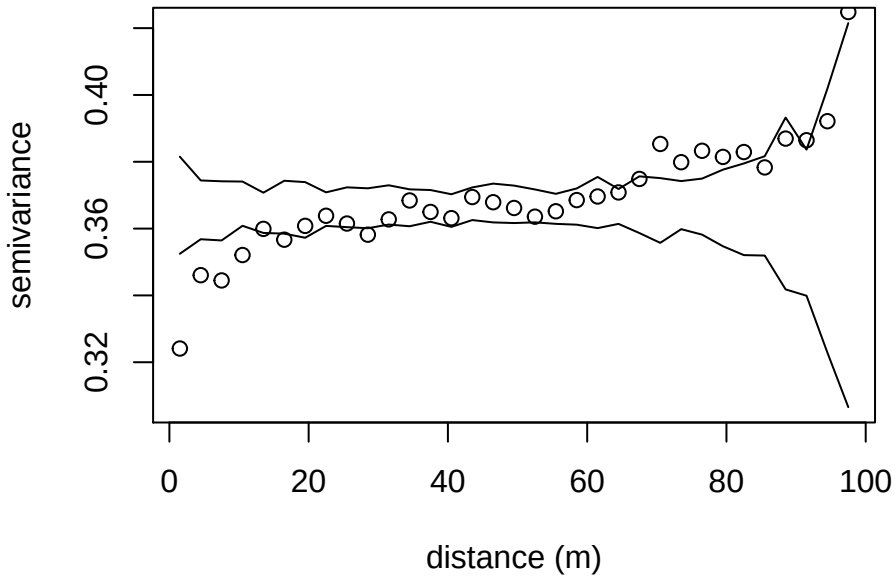


The variogram indicates, in this case, a clear spatial genetic structure :

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

# compute statistical envelope
env <- randsvariog(var=va, X=larix2300, nsim=30, bounds=c(0.025, 0.975),
  save.sim=FALSE)
```

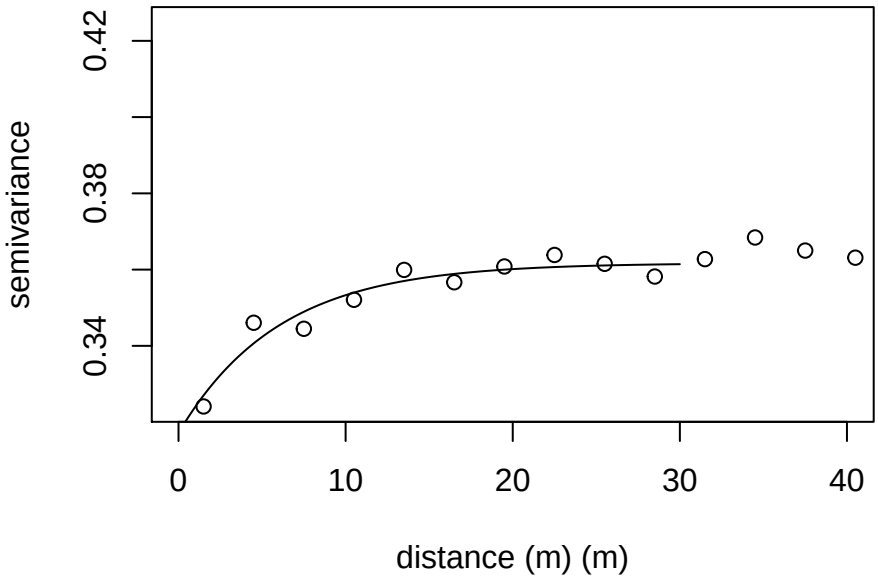
```
## ..............................
## done
```

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



Because the variogram reveals a spatial genetic structure, fitting an exponential model will provide estimates of various parameters synthesizing the SGS at hand. Since the variogram reaches a plateau at ca. 30 m but rises again for distances > 60 m, it is useful to restrain the distance range used in the fitting procedure. This is achieved by setting `max.dist` to 40 m.

```
# fit exponential model to the empirical variogram
fit <- fitsvariog(vario=va, ini.cov.pars=c(0.1,20), nugget=0.3,
  max.dist=30, plot = FALSE)

# plot results
plot(va$svario$u, va$svario$v, xlim=c(0, 40),
    xlab="distance (m) (m)", ylab="semivariance")
lines(fit$fit)
```



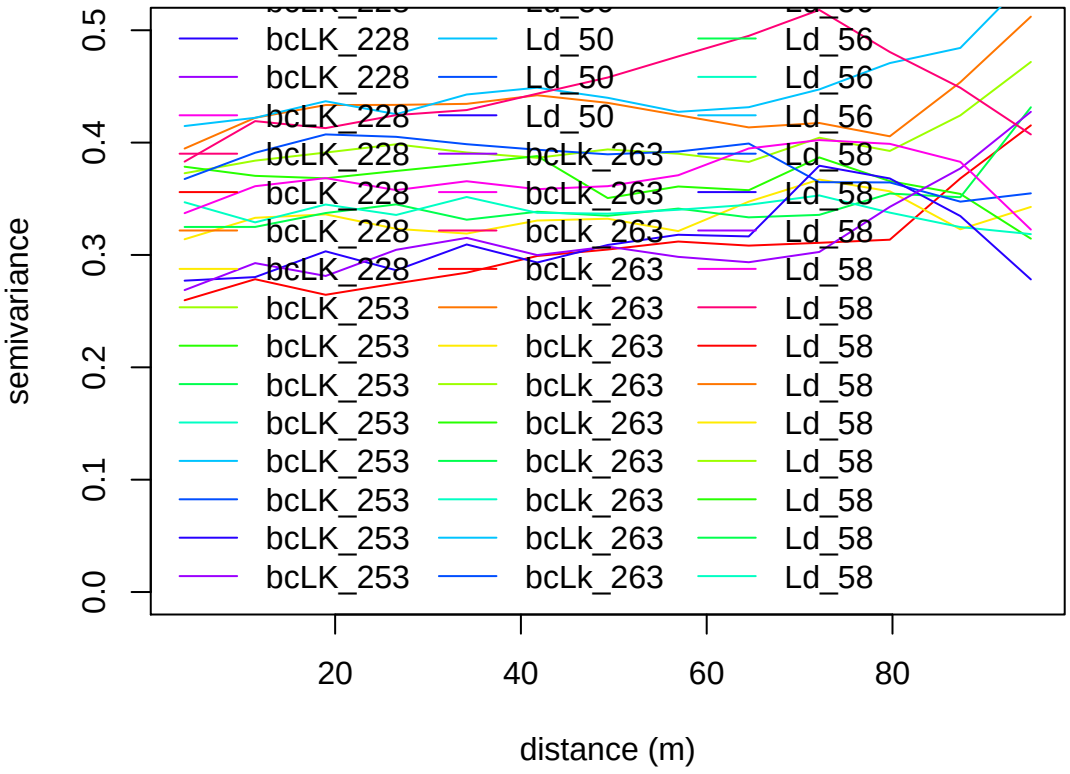
The estimate of the parameters are given in:

```
fit$param
```

```
##           c1    nugget    range pract.range     sill      Hhat        FN
## 1 0.04455914 0.3171648 6.007748    18.02324 0.361724 0.3663014 0.1231855
##           bf        Sp
## 1 -0.1664517 0.1898369
```

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 locus-by-locus semivariances are 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 (m)",
  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)
```



2.5 sim01, sim02 and sim03

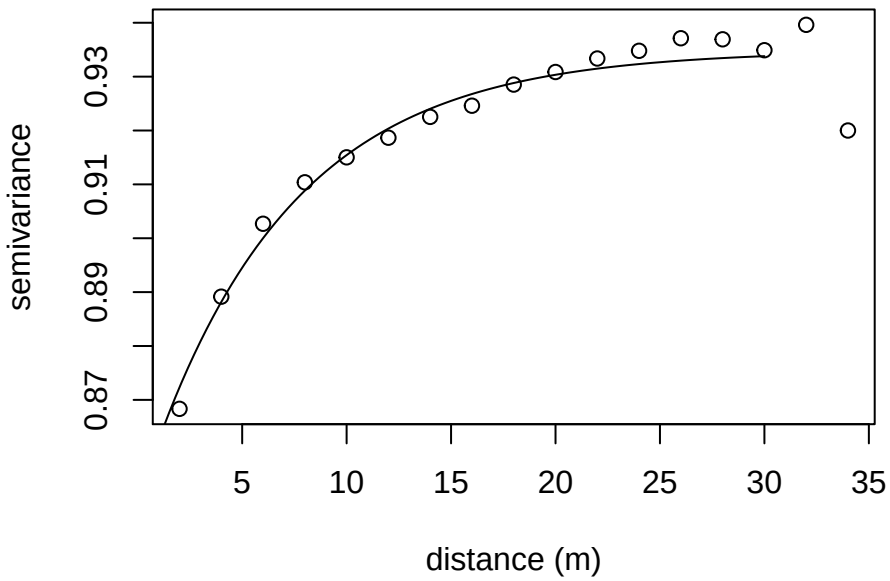
These datasets were generated by simulation with the software IBDsim (Leblois et al 2009). - `sim01` corresponds to data under SMM generated using the following parameters: 625 gene copies, 20 loci, 25 x 25 haploid individuals evolving at G=0 on a 300 x 300 lattice with absorbing boundaries, mutation proba=0.001, Mrca Moy= 286842, MRCA MAX=617227. - `sim02` was constructed by taking the first 100 individuals from `sim01` and modifying their spatial coordinates so as to create a clumped distribution and randomizing the genotypes (thus removing the spatial genetic structure). - `sim03` was constructed by taking the first 100 individuals from `sim01`.

```
data(sim01)
# 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, xlab="distance (m)", ylab="semivariance")

# fit exponential model to the empirical variogram
fit <- fitsvariog(vario=va, ini.cov.pars=c(0.5,20), nugget=8, max.dist=30,
                  plot = FALSE)
fit$param
```

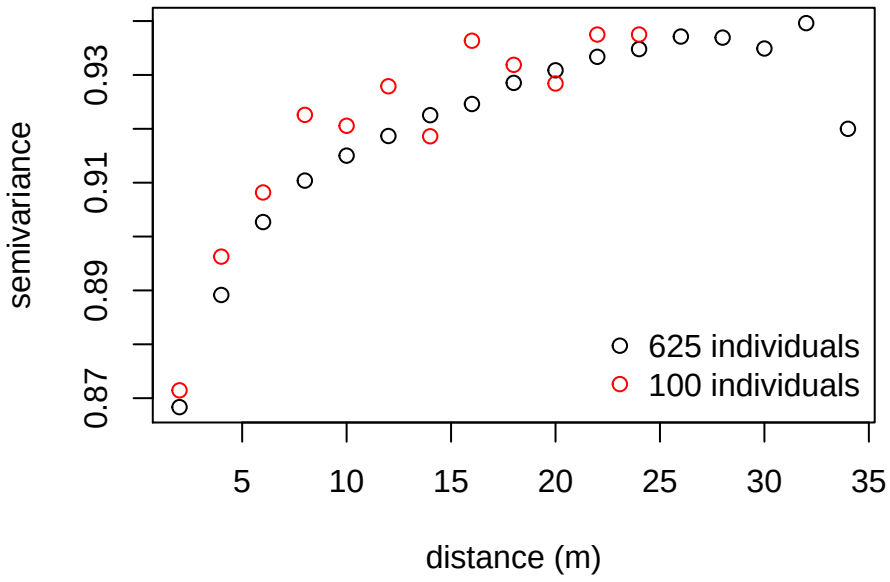
```
##           c1    nugget     range pract.range      sill      Hhat          FN
```

```
## 1 0.08374101 0.8510982 6.838841    20.51652 0.9348392 0.9173031 0.08957798
##            bf        Sp
## 1 -0.1462236 0.1606108
```

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

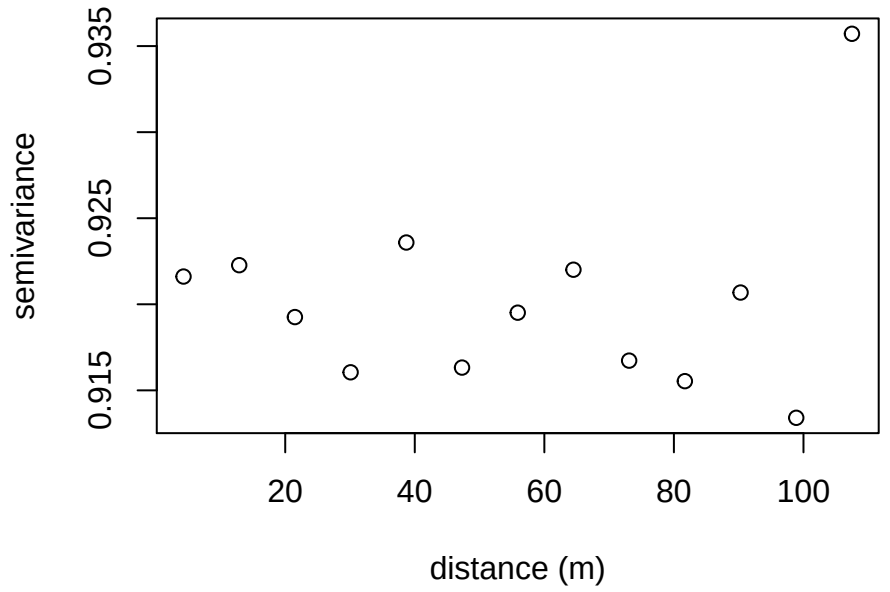



```
data(sim03)
# compute variogram
va3 <- svariog(X=sim03, uvec=distlag(dist=sim03$coord, dmin=1,
  distance.lag=2), plot=FALSE)

plot(va$svario$u, va$svario$v, xlab="distance (m)", ylab="semivariance")
points(va3$svario$u, va3$svario$v, col="red")
legend("bottomright", legend=c("625 individuals", "100 individuals"),
  col=c("black", "red"), bty="n", lty=0, pch=1)
```



This illustrates that the spatial signal becomes more erratic when the number of sampled individuals decreases: sampling effort is a very important issue when it comes to spatial analyses.

```
data(sim02)
va2 <- svariog(X=sim02, plot=FALSE)
plot(va2$svario$u, va2$svario$v, xlab="distance (m)", ylab="semivariance")
```



The variogram shows that there is no spatial genetic structure in dataset `sim02`.

2.6 Wagner

This dataset is the toy example used in Wagner et al. (2005) p. 1750-1752.

```
data(Wagner)

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

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

```
mat <- genoweight(X=Wagner,genotypes=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
```

Individuals 1 and 2 are similar and thus the weight of this couple is 0.

If we compute the variogram for genetic diversity and look at the distance classes, we see that the third distance class corresponding to a lag of 3 distance units is omitted. This is because it involves only one data pair:

```
wa <- varioWeight(X=Wagner, weights=mat, uvec=c(1,2,3))
```

```
## as.geodata: 2 replicated data locations found.
##  Consider using jitterDupCoords() for jittering replicated locations.
## WARNING: there are data at coincident or very closed locations, some of the geoR's fu
##  Use function dup.coords() to locate duplicated coordinates.
##  Consider using jitterDupCoords() for jittering replicated locations
## variog: co-locatted data found, adding one bin at the origin
```

```
wa$svario$u # corresponds distance r in Wagner et al 2005 p 1751
```

```
## [1] 0 1 2
```

```
wa$svario$gamma # raw semivariances corrsponding to Hhat(r)
```

```
## [1] 0.2500 0.4375 0.8750
```

```
#in Wagner et al 2005 p 1751
wa$svario$n # corresponds distance nr in Wagner et al 2005 p 1751
```

```
## [1] 2 8 4
```

Note that `varioWeight` issues some warning messages because some points are co-located. Finally, note also that `wa$svario$v` is the weighted semivariance for geneotypic diversity and should not be mistaken for the values reported for the molecular variance in Wagner et al (2005) p. 1752.

3 Aknowledgements

The INRA department EFPA provided financial support to a prehistoric 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.

References

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.

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

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.

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.