Topic 1: Covariance functions and variograms modelling

In this topic we continue to study covariance functions and variograms. In particular, we consider

- Variogram's **parameter estimation**.
- Example of **fitting variograms to simulated data.**
- Example of **fitting variograms to the Meuse data.**
- **Anisotropy**.

geoR variogram's parameter estimation.

The traditional way of finding a suitable covariance function (variogram) model to data is to fit a parametric model to a sample covariance function (variogram).

For fitting a covariance function model to the sample covariance function, one uses these steps:

- **Visual inspection** of a sample covariance function (variogram) and **choosing a suitable theoretical model** (such as exponential, Gaussian, Matérn, ...), with or without nugget.
- **Choosing suitable initial values** for model's parameters: variance, range, scale, ..., and possibly nugget.
- **Fitting the model** by using one of the fitting criteria.

To see the list of covariance models available in GEOR type:

```
> library(geoR)
> ?cov.spatial
```

Details

Covariance functions return the value of the covariance *C(h)* between a pair variables located at points separated by the distance *h*. The covariance function can be written as a product of a variance parameter *sigma^2* times a positive definite *correlation function rho(h)*:

C(h) = sigma^2 * rho(h).

The expressions of the covariance functions available in **geoR** are given below. We recommend the *LaTeX* (and/or the corresponding *.dvi*, *.pdf* or *.ps*) version of this document for better visualization of the formulas.

Denote *phi* the basic parameter of the correlation function and name it the *range parameter*. Some of the correlation functions will have an extra parameter *kappa*, the *smoothness parameter*. K_κ(x) denotes the modified Bessel function of the third kind of order *kappa*. See documentation of the function besselK for further details. In the equations below the functions are valid for *phi > 0* and *kappa > 0*, unless stated otherwise.

cauchy

rho(h) = [1+(h/phi)^2]^(-kappa)

gencauchy (generalised Cauchy)

rho(h) = [1+(h/phi)^kappa2]^(-kappa1/kappa2), kappa1 > 0, 0 < kappa2 <=0

circular
Let *theta = min(h/phi,1)* and

gamma(h)= 2 * ((θ * √{1-θ^2} + \sin^{-1} √{theta}))/pi.

Then, the circular model is given by:

rho(h) = 1 - gamma(h) if h < phi , 0 otherwise

cubic

rho(h) = 1 - (7 * ((h/phi)^2) - 8.75 * ((h/phi)^3) + 3.5 * ((h/phi)^5) - 0.75 * ((h/phi)^7)) if h < phi , 0 otherwise.

gaussian

Parameters of the selected model can be estimated by several methods. For example, the following two approaches are very popular in the majority of applications:

- **ordinary least squares** fit (option **OLS**) of empirical variograms, by using the function VARIOFIT;
- **maximum likelihood methods** (option **ML**), by using the function LIKFIT.

In the parameter estimation functions **variofit** and **likfit** the nugget effect parameter can either be estimated or set to a fixed value. The same applies for smoothness, anisotropy and transformation parameters.

Options for taking trends into account are also included. Trends can be specified as polynomial functions of the coordinates and/or linear functions of given covariates.

Example 1.

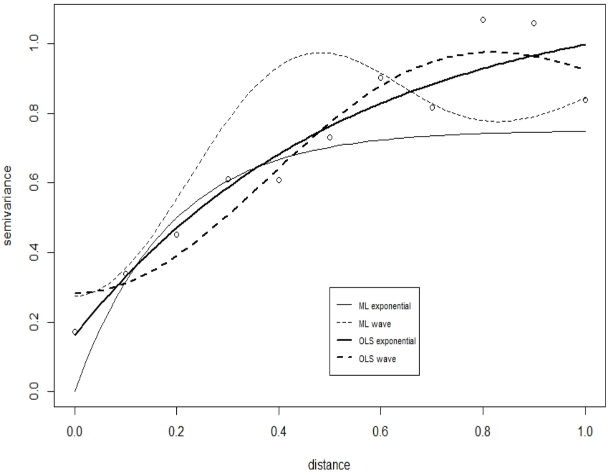
We will use the simulated data set s100 considered before.

The commands below show exponential and wave models fitted by the ML and OLS methods with the option for the estimated nugget parameter.

```
> data(s100)
> ml.n <- likfit(s100, ini = c(1,0.5),  nug = 0.5,
+ cov.model = "exponential")
> ml1.n <- likfit(s100, ini = c(1,0.5),  nug = 0.5,
+ cov.model = "wave")

> bin1 <- variog(s100, uvec=seq(0,1,l=11))
> ols.n <- variofit(bin1, ini = c(1,0.5),nugget=0.5,
+ weights="equal", cov.model = "exponential")
> ols1.n <- variofit(bin1,ini = c(1,0.5),nugget=0.5,
+ weights="equal", cov.model = "wave")
```

Now, to plot the fitted models against the empirical variogram one can use the following commands:

```
> plot(bin1)

> lines(ml.n, max.dist = 1)
> lines(ml1.n,lty = 2, max.dist = 1)
> lines(ols.n, lwd = 2, max.dist = 1)
> lines(ols1.n, lty = 2, lwd = 2, max.dist = 1)

> legend(0.5, 0.3, legend=c("ML exponential",
+ "ML wave","OLS exponential","OLS wave"),
+ lty=c(1,2,1,2),lwd=c(1,1,2,2), cex=0.7)
```



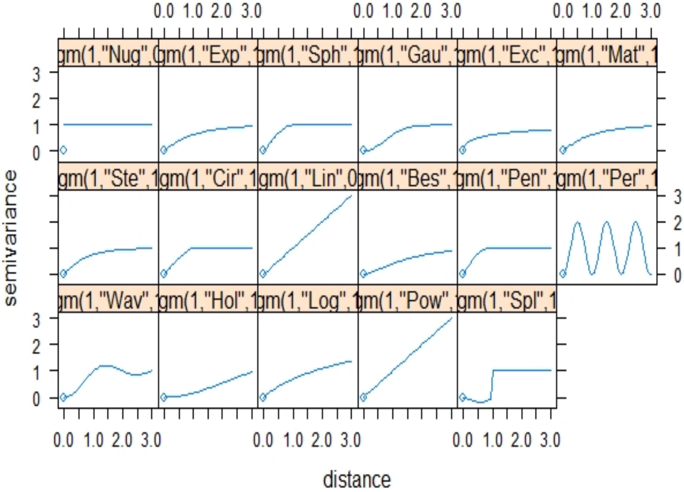
To summarize the fitted models we can use short or more detailed summaries. For example, for the exponential model with estimated nugget fitted by the ML an OLS, type the commands:

```
> ols.n
variofit: model parameters estimated by OLS (ordinary
least squares):
covariance model is: exponential
parameter estimates:
tausq sigmasq     phi
0.1619  0.9868  0.5325
Practical Range with cor=0.05 for asymptotic range: 1.595346
variofit: minimised sum of squares = 0.0711

> summary(ml.n)
> summary(ols.n)
```

Fitting variograms for Meuse data.

To see the list of variograms available in **gstat** and their plots type:

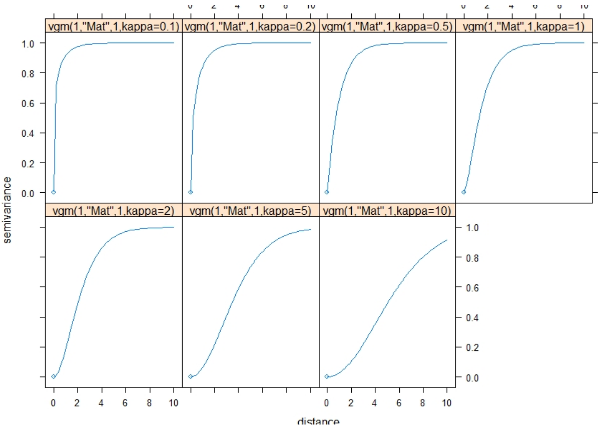
```
> library(gstat)
> show.vgms()
> vgm()
short                                           long
1     Nug                                  Nug (nugget)
2     Exp                             Exp (exponential)
3     Sph                               Sph (spherical)
4     Gau                                Gau (gaussian)
5     Exc        Exclass (Exponential class/stable)
6     Mat                                  Mat (Matern)
7     Ste Mat (Matern, M. Stein's parameterization)
8     Cir                                Cir (circular)
9     Lin                                  Lin (linear)
10    Bes                                  Bes (bessel)
11    Pen                           Pen (pentaspherical)
...
```



gm(1,"Nug",0
gm(1,"Exp",1
gm(1,"Sph",1
gm(1,"Gau",1
gm(1,"Exc",1
gm(1,"Mat",1
gm(1,"Ste",1
gm(1,"Cir",1
gm(1,"Lin",0
gm(1,"Bes",1
gm(1,"Pen",1
gm(1,"Per",1
gm(1,"Wav",1
gm(1,"Hol",1
gm(1,"Log",1
gm(1,"Pow",1
gm(1,"Spl",1
semivariance
distance
0.0 1.0 2.0 3.0
0 1 2 3

To produce several plots of a specific variogram (in this case Matern) for different parameters (in this case kappa.range) one can type:

```
> show.vgms(model = "Mat", kappa.range = c(0.1, 0.2, 0.5,
+ 1, 2, 5, 10), max = 10)
```



Example 2.

For weighted least squares fitting a variogram model to the sample variogram, we need initial values for the variogram fit, because for many models fitting parameters involves non-linear regression.

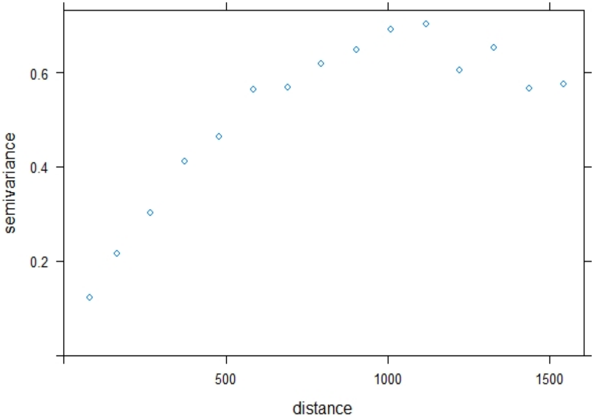
For example, let us study the meuse data again. The following fit with the spherical model works:

```
> library(lattice)
> library(sp)
> data(meuse)
> coordinates(meuse) <- c("x", "y")
> v  <- variogram(log(zinc) ~ 1, meuse)
```

In the last formula, the ~ 1 defines a single constant predictor, leading to a spatially constant mean.

First let us plot the sample variogram:

```
> plot(v)
```



The spherical model looks like a reasonable choice.

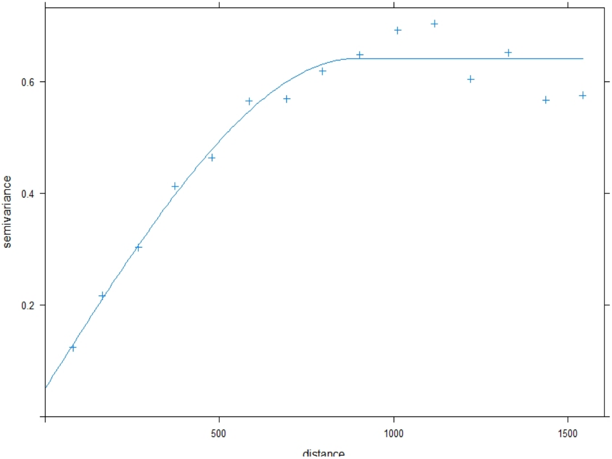
However if we choose initial values too far off from reasonable values, the fit will not succeed:

```
> fit.variogram(v, vgm(1, "Sph", 10, 1))
model psill range
1   Nug     1     0
2   Sph     1    10
Warning message:
In fit.variogram(v, vgm(1, "Sph", 10, 1)) :
singular model in variogram fit
```

A better selection of the initial values results in a properly fitted model:

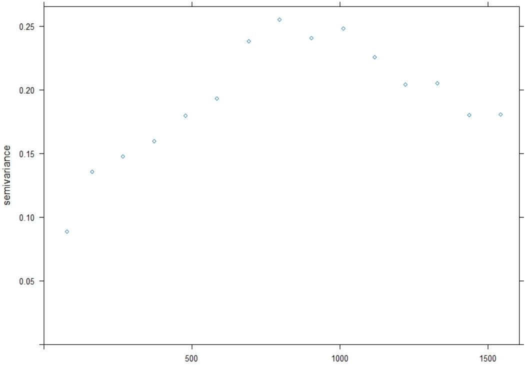
```
> fit.variogram(v, vgm(1, "Sph", 800, 1))
model      psill    range
1   Nug 0.05065923   0.0000
2   Sph 0.59060463 896.9976
```

```
> v.fit <- fit.variogram(v, vgm(1, "Sph", 800, 1))
> plot(v, v.fit, pch = 3)
```



Residual variograms are calculated by default when more complex models for trends are used, for example as in

```
> variogram(log(zinc) ~ sqrt(dist), meuse)
np       dist      gamma dir.hor dir.ver   id
1   57   79.29244 0.08819594       0       0 var1
2  299  163.97367 0.13523671       0       0 var1
3  419  267.36483 0.14718465       0       0 var1
4  457  372.73542 0.15929716       0       0 var1
5  547  478.47670 0.17933406       0       0 var1
6  533  585.34058 0.19298151       0       0 var1
7  574  693.14526 0.23756378       0       0 var1
...

> plot(variogram(log(zinc) ~ sqrt(dist), meuse))
```



Here, the trend is defined by the model

$$\log(Z(s)) = \beta_0 + \sqrt{D(s)}\beta_1 + \varepsilon(s),$$

with the distance to the river $D(s)$, and the residuals $\varepsilon(s)$.

Anisotropy

Anisotropy may be modelled by defining a range ellipse instead of a circular or spherical range. In the following example we fit first the isotropic model:

```
> ml3.n <-  likfit(coords =coordinates(meuse),
+ data =log(meuse$zinc), ini = c(1,800), nug = 0.05,
+ cov.model = "spherical")

> ml3.n
likfit: estimated model parameters:
beta      tausq    sigmasq        phi
"  6.0524" "  0.0244" "  0.5711" "852.4799"
Practical Range with cor=0.05 for asymptotic range: 852.4799

likfit: maximised log-likelihood = -100.7
```

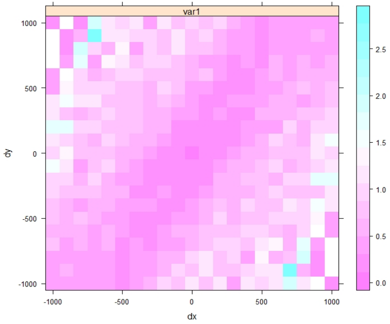
and then the anisotropic one:

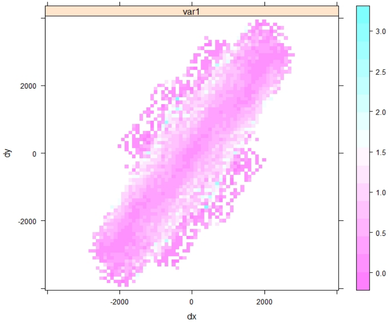
```
> ml4.n <-  likfit(coords =coordinates(meuse),
+ data =log(meuse$zinc), ini = c(1,800), fix.psiA = FALSE,
+ fix.psiR = FALSE, nug = 0.05, cov.model = "spherical")

> ml4.n
likfit: estimated model parameters:
beta      tausq    sigmasq        phi        psiA       psiR
"  6.0069" "  0.0040" "  0.5978" "728.4558" "  0.4942" "  2.3375"
Practical Range with cor=0.05 for asymptotic range: 728.4558

likfit: maximised log-likelihood = -91.84
```

When enough measurements are available, one may consider plotting a variogram map, applying the commands

```
> plot(variogram(log(zinc) ~ 1, meuse, map = TRUE,
+ cutoff = 1000, width = 100))
```

```
> plot(variogram(log(zinc) ~ 1, meuse, map = TRUE,
+ cutoff = 4000, width = 100))
```





In this case vectors **h** binned in square grid cells over x and y. In the second plot larger distances are used which let to see the spatial pappern in the variogram in much more detail.

The both plots suggest elliptic structure, i.e. anisotropy.

Key R commands

gstat	*package for spatial geostatistical modelling, prediction and simulation*
likfit(geodata,...)	*maximum likelihood (ML) parameter estimation*
variog(x)	*computes sample (empirical) variograms*
variofit(vario,...)	*fits a parametric model to a empirical variogram*
show.vgms(model,...)	*plots a range of variogram models*
vgm()	*prints key information of variogram models*
variogram(object, ...)	*calculates the sample variogram*
fit.variogram(object, model)	*fits a variogram model to a sample variogram*