LAB Geostatistics

```
## Load spatial packages
library(sp)          ## Data management
library(lattice)     ## Data management
library(geoR)        ## Geostatistics
```

```
## ------------------------------------------------------------
##  Analysis of Geostatistical Data
##  For an Introduction to geoR go to http://www.leg.ufpr.br/geoR
##  geoR version 1.8-1 (built on 2020-02-08) is now loaded
## ------------------------------------------------------------
```

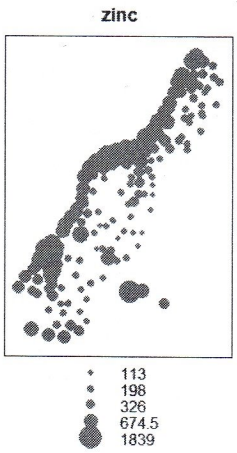
```
library(gstat)       ## Geostatistics

## Functions for graphics
v.f     <- function(x, ...){100-cov.spatial(x, ...)}
v.f.est <- function(x,C0, ...){C0-cov.spatial(x, ...)}

### ----------------------------------------------------------------------------
### ----------------------------------------------------------------------------
### EXPLORATORY ANALYSIS & VARIOGRAM ESTIMATION
### ----------------------------------------------------------------------------
### ----------------------------------------------------------------------------
## Load meuse data set:
## The meuse is a classical geostatistical data set used frequently
## to demonstrate various geostatistical analysis steps.
## The point data set consists of 155 samples of top soil heavy metal
## concentrations (ppm), along with a number of soil and landscape variables.
## The samples were collected in a flood plain of the river Meuse,
## near the village Stein (The Netherlands).

data(meuse)

## Define the sample coordinates
coordinates(meuse) <- c('x','y')

# bubble plot(obj,zcol,...)
# key.space=location of the key
bubble(meuse,'zinc',do.log=TRUE,key.space='bottom')
```

in this way R understand what dataset it is

meuse :

x	y	features	dist	...

distance from the river "Meuse"



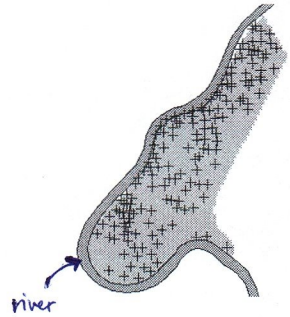

```
# river meuse
data(meuse.riv)
meuse.lst <- list(Polygons(list(Polygon(meuse.riv)), "meuse.riv"))
meuse.sr <- SpatialPolygons(meuse.lst)

# grid for prediction
data(meuse.grid)
is(meuse.grid)
```

```
## [1] "data.frame" "list"       "oldClass"   "vector"
```

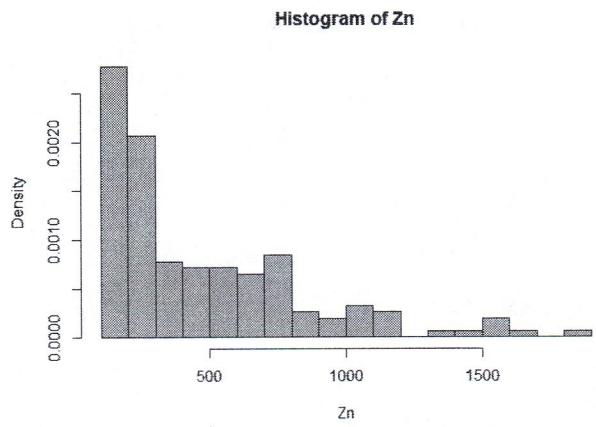
We have to translate these in a format that is geographical

```
coordinates(meuse.grid) <- c('x','y')
meuse.grid <- as(meuse.grid, 'SpatialPixelsDataFrame')

# plot all together
image(meuse.grid, col = "lightgrey")
plot(meuse.sr, col = "grey", add = TRUE)
plot(meuse, add = TRUE)
title('meuse river geostatistical data')
```

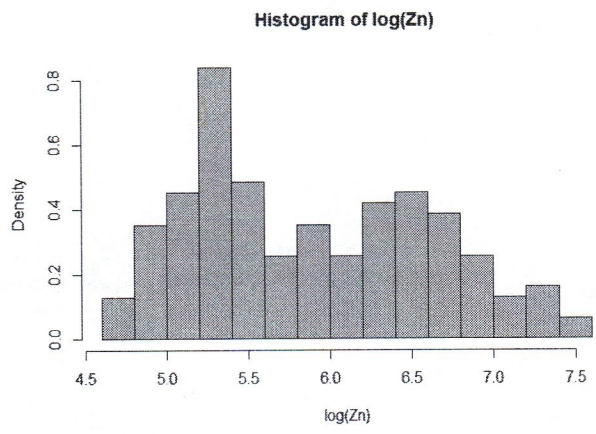
meuse river geostatistical data




```
### ------------------------------------------------------------------------
### Exploratory Analysis
### ------------------------------------------------------------------------

# histogram of zinc variable
hist(meuse$zinc, breaks=16, col="grey", main='Histogram of Zn', prob = TRUE, xlab = 'Zn')
```




```
# highly skewed, transform to the log
hist(log(meuse$zinc), breaks=16, col="grey", main='Histogram of log(Zn)', prob = TRUE, xlab = 'log(Zn)')
```



Note: gaussianity is not needed but a lot of geostatistical methods work better under the gaussian assumptions

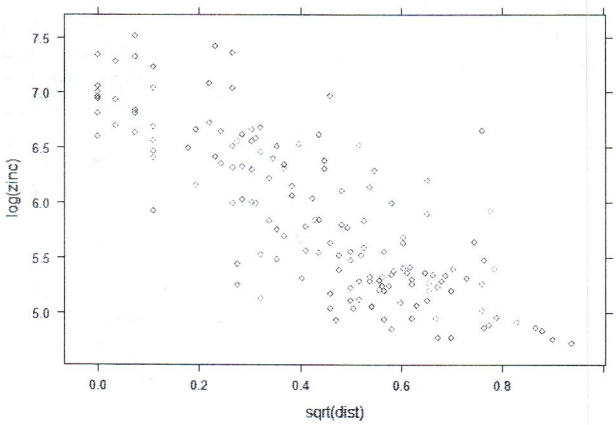
(for example: **kriging** is not developed under gaussianity but we have seen that if the data are gaussian we have some nice properties, in particular if the data are gaussian the kriging predictor will be our approximation of the conditional expectation (which is the best prediction according to our criteria))

```
# scatterplot of log(zinc) with respect to distance from the river
xyplot(log(zinc) ~ sqrt(dist), as.data.frame(meuse))
```

→ we're checking if log(zinc) has a geographical dependence but with the geographical analysis we can't say what is influencing this dependence

⟹ we have a variable which tells us the distance from the river, so if we see a correlation between log(zinc) and the distance then maybe the river could be the explanation of the geographical dependence

this in the case of STATIONARY MODEL

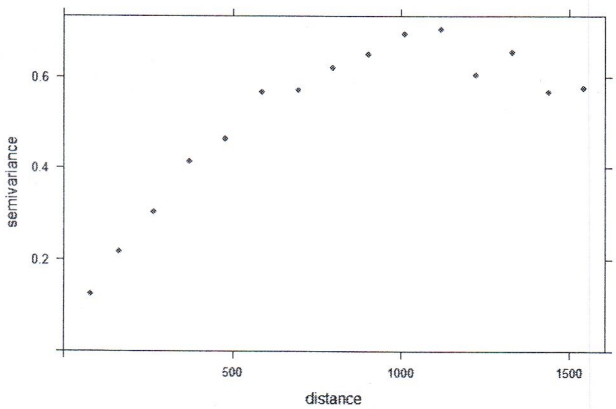


it seems to be an association between the two (negative correlation ⇒ closer to the river it seems to be a higher concentration of the zinc)

```
# Negative correlation: lower distance from the river => higher level of zinc

### ------------------------------------------------------------------------
### Estimating Spatial Correlation Variogram Analysis
### ------------------------------------------------------------------------

# sample variogram (binned estimator)
svgm <- variogram(log(zinc) ~ 1, meuse)
plot(svgm, main = 'Sample Variogram',pch=19)
```

← variogram for estimation the structure of the spatial dependence

~ 1 if we want to use a stationary model (in a stationary model we don't use any covariate)

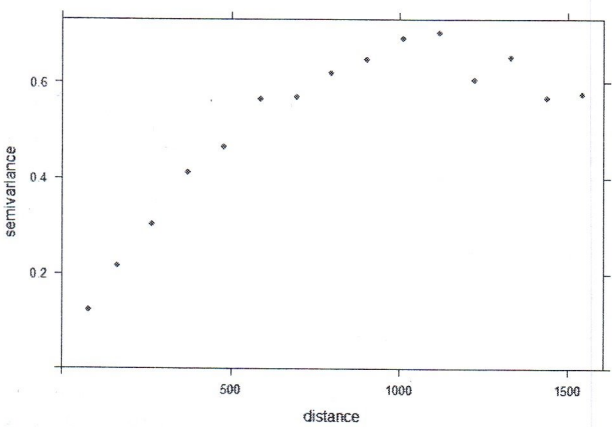


How these points are built:
we divide the domain of distances in classes and for each class of distance we take the pairs of data whose distance is in the class that we're considering; we look at their square distances, we take an average and we plot the point according to the average

it look like a stationary variogram

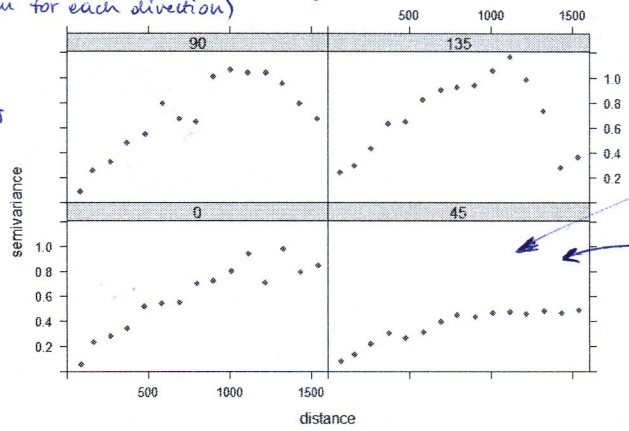
```
# N.B.  the notation "~ 1" stands for a single constant predictor
#       (hp: spatially constant mean)

# default decisions:
# direction dependence, cutoff, lag width

# the following
plot(variogram(log(zinc) ~ 1, meuse),pch=19)
```

→ by default R set isotropic variograms, if we want to use an anisotropic variogram we have to specify some angles




```
# automatically decides to ignore direction: point pairs are merged on the
# basis of distance to compute the empirical variogram

plot(variogram(log(zinc) ~ 1, meuse, alpha = c(0, 45, 90, 135)),pch=19)
```

In each panel we have an angle:
(one variogram for each direction)



If we have isotropy we should see more or less the same variogram in all the panels

this direction is parallel (*) to most part of the river, where we saw that in fact we had a kind of pattern of the data that was following the river (so this low variability is not random)

this variogram seems to be different: there is a different asymptote
⟹ lower variability of data in direction $\alpha = 45°$
⟹ ZONAL ANISOTROPY (which may be due to non stationarity of the field)

We can:
- go on (*)
- model an anisotropic structure
- model a drift term and recover a residual which is isotropic

(*) since the low variability is basically due to the river we go on and consider the model isotropic

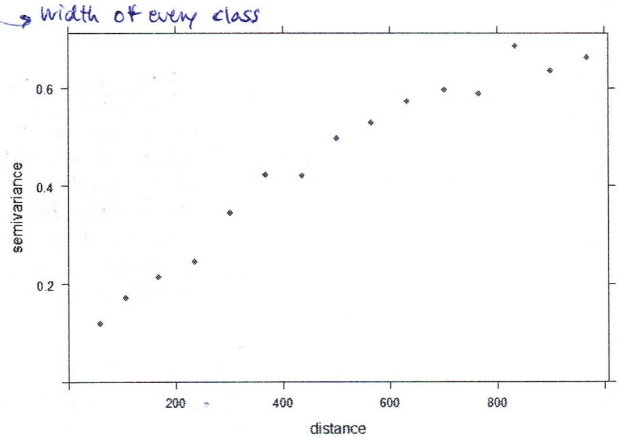
```
# point pairs whose separation vector has a given direction are used in each
# panel (not too many directions otherwise noise will increase)
# Note: zonal anisotropy

# cutoff distance: maximum distance up to which point pairs are considered
#                  (default = bbox diagonal / 3)
# Lag width: width of distance intervals over which point pairs are averaged
#            in bins (default = cutoff distance / 15)

plot(variogram(log(zinc) ~ 1, meuse, cutoff = 1000, width = 1000/15),pch=19)
```

In this case R put 1500 as cutoff, but we can modify it. It's not a good idea to use all the data that we have and all the pairs that we have at all the distances. Why? The higher the distance the higher the variability of estimating ⇕ (we could end up with a very uncertain estimator)

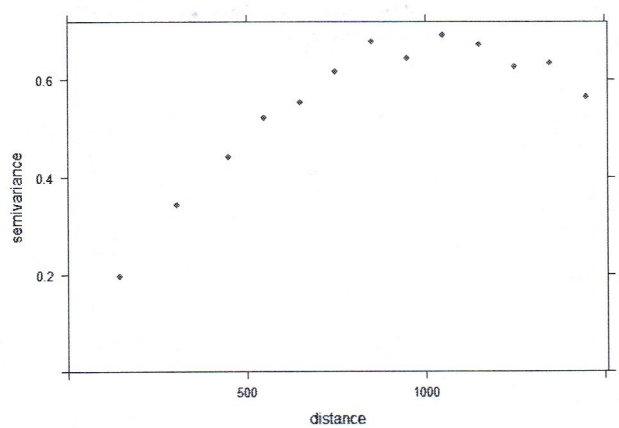
⇓

the cutoff must be much lower than the higher distance that we have between data

width of every class




```
# intervals can have different widths: to fix varying widths use the argument boudaries
plot(variogram(log(zinc) ~ 1, meuse, boundaries = c(0,200,seq(400,1500,100))),pch=19)
```



NOTE:
too many or too few classes can be a problem

```
# useful for data sets that have much information on short distance variability

### ---------------------------------------------------------------------
### Variogram modeling
### ---------------------------------------------------------------------

# list of parametric isotropic variogram models
vgm()
```

Now that we have our EMPIRICAL VARIOGRAM we have to model (with a parametric family) the variogram
- choose a parametric family
- fit the variogram

```
##    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)
## 12   Per                       Per (periodic)
## 13   Wav                          Wav (wave)
## 14   Hol                          Hol (hole)
## 15   Log                    Log (logarithmic)
## 16   Pow                         Pow (power)
## 17   Spl                        Spl (spline)
## 18   Leg                      Leg (Legendre)
## 19   Err            Err (Measurement error)
## 20   Int                     Int (Intercept)
```

pure nugget effect (no spatial dependence)

both linear at 0 (most famous)

quadratic near 0 (good for geneval data, bad for simulated data)

very flexible (most used)

Remember: we can combine them (we can use nested families)

```
# in gstat, valid variogram models are constructed by using one or
# combination of two or more basic variogram models
# first argument of the function 'vgm' is partial sill,
# then the desired model, then range, and finally nugget:
# vgm(sill, model, range, nugget)

# some examples...
vgm(1, "Sph", 300)
```

```
##   model psill range
## 1   Sph     1   300
```

Spherical model

```
vgm(1, "Sph", 300, 0.5)
```

```
##   model psill range
## 1   Nug   0.5     0
## 2   Sph   1.0   300
```

Spherical model with nugget (we're giving 2 structures)

```
# one can also add two or more models
v1 <- vgm(1, "Sph", 300, 0.5)
v2 <- vgm(0.8, "Sph", 800, add.to = v1)
v2
```

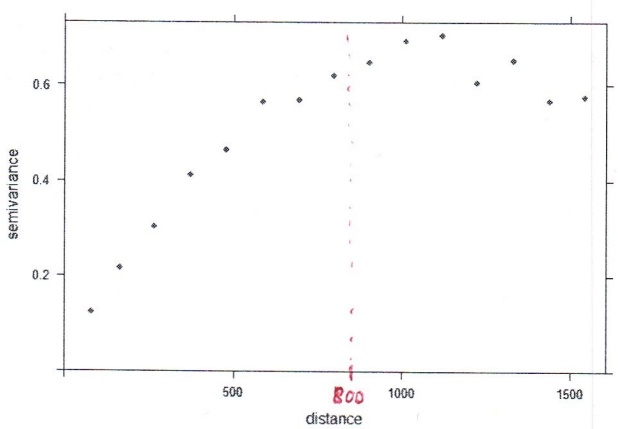
```
##   model psill range
## 1   Nug   0.5     0
## 2   Sph   1.0   300
## 3   Sph   0.8   800
```

Nested model

```
# this is only measurement error
vgm(0.5, "Nug", 0)
```

```
##   model psill range
## 1   Nug   0.5     0
```

```
## weighted least squares fitting a variogram model to the sample variogram
## STEPS:
## 1) choose a suitable model
## 2) choose suitable initial values for partial sill, range & nugget
## 3) fit the model using one of the possible fitting criteria

v <- variogram(log(zinc) ~ 1, meuse)

plot(v,pch=19)
```




```
# Linear behavior near the origin, growth not very fast
# Recall: both spherical and exponential model have a linear behavior near the
#         origin but exponential model has a faster growth than the spherical one
# => we fit a spherical model

# try reasonable initial values
fit.variogram(v, vgm(1, "Sph", 800, 1))
```

!!

```
##   model     psill    range
## 1   Nug 0.05065923   0.0000
## 2   Sph 0.59060463 896.9976
```

value of the range (the value of the distance at which the variogram stabilizes)

~0.65

final asymptote (sum of the two)

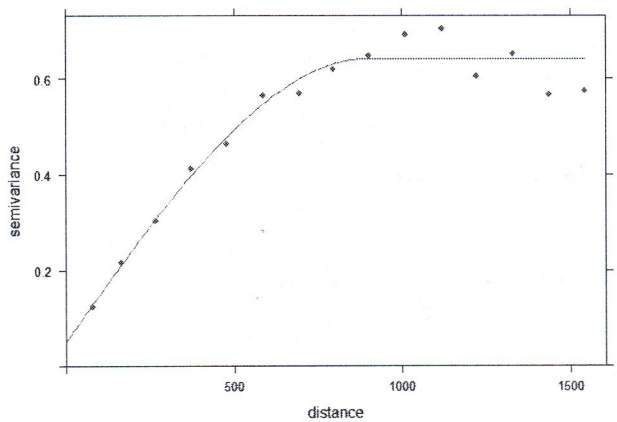
```
# try unreasonable initial values
fit.variogram(v, vgm(1, "Sph", 10, 1))
```

this is the error which says that we're doing ~~so~~ BAD the fitting of the variogram

```
## Warning in fit.variogram(v, vgm(1, "Sph", 10, 1)): singular model in variogram
## fit
```

```
##   model psill range
## 1   Nug     1     0
## 2   Sph     1    10
```

```
# due to high non linearity in the minimization problem,
# starting from unreasonable initial values might cause fail to converge

# plot of the final fit
v <- variogram(log(zinc) ~ 1, meuse)
v.fit <- fit.variogram(v, vgm(1, "Sph", 800, 1))
plot(v, v.fit, pch = 19)
```




```
# fitting method: non linear regression with minimization of weighted
# sum of squares error. final value of the minimum
attr(v.fit, 'SSErr')
```

```
## [1] 9.011194e-06
```

```
# how can we choose weights? argument fit.method in fit.variogram
# fit.method = 1 : w = N_j
# fit.method = 2 : w = N_j/gamma(h_j)^2
# fit.method = 6 : w = 1
# fit.method = 7 : w = N_j/h_j^2

# one can also keep one of the parameters fixed, and fit only the others.
# this is common for the nugget parameter, which may be hard to infer from data
# when sample locations are regularly spread. Information may be derived from
# measurement error characteristics for a specific device.

# ex: fix the nugget variance to the value 0.06
fit.variogram(v, vgm(1, "Sph", 800, 0.06), fit.sills = c(FALSE, TRUE))
```

we're fixing by hand some of the parameters

```
##   model     psill    range
## 1   Nug 0.0600000   0.0000
## 2   Sph 0.5845836 923.0066
```

```
# the range parameters can be fixed using argument fit.ranges

## maximum likelihood fitting of variogram models
## - does not need the sample variogram
## - can be performed through restricted maximum likelihood

fit.variogram.reml(log(zinc)~1, meuse, model=vgm(0.6, "Sph", 800, 0.06))
```

if we use a maximum likelihood we have to impose a distribution (/likelihood) [gaussian tipically]

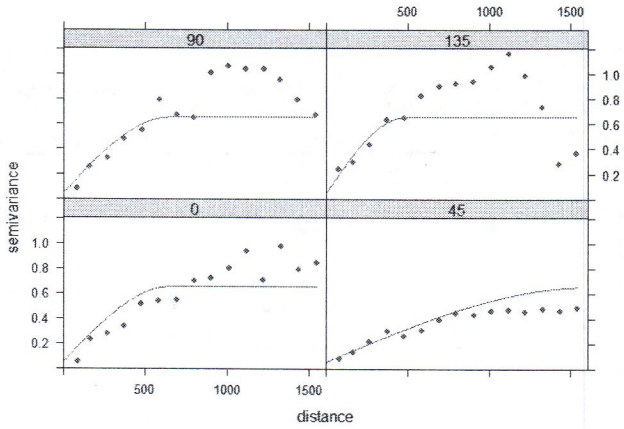
```
##   model      psill range
## 1   Nug 0.02006905     0
## 2   Sph 0.57134920   800
```

```
v.fit
```

```
##   model      psill    range
## 1   Nug 0.05065923   0.0000
## 2   Sph 0.59060463 896.9976
```

```
## modeling anisotropy*
v.dir <- variogram(log(zinc)~1,meuse,alpha=(0:3)*45)
v.anis <- vgm(.6, "Sph", 1600, .05, anis=c(45, 0.3))

print(plot(v.dir, v.anis, pch=19))
```

↳ we can give the main direction of anisotropy




```
###
###
### SPATIAL PREDICTION & KRIGING
###
###

###
### Stationary Univariate Spatial Prediction (Ordinary Kriging)
###

## Prediction in a single new location
s0.new=data.frame(x=179180, y=330100)
coordinates(s0.new)=c('x','y')

# plot all together
image(meuse.grid, col = "lightgrey")
plot(meuse.sr, col = "grey", add = TRUE)
plot(meuse, add = TRUE)
plot(s0.new, add = TRUE, col='red', lwd = 2)
title('meuse river geostatistical data')
```

(let's assume that our field is isotropic and stationary)

Kriging:
- choose a new location
- make prediction in this new location

we have to be careful on how we build the data frame; we have to give the names of the columns equal to the ones in the training dataset (same name and same structure)

NOTE that these are UTM coordinates, not lat/long, these are the right coordinates for the estimation of variogram and prediction (it's about the distances)

(the variogram uses euclidean distances, UTM can be good but long/lat use angles so the euclidean distance is not good)

ALWAYS TRANSLATE IN **UTM**


```
# Create a gstat object setting a spherical (residual) variogram
# gstat(g.obj, id, formula, data, model, set,...)
g.tr <- gstat(formula = log(zinc) ~ 1, data = meuse, model = v.fit)

## ordinary kriging
# Make the ordinary kriging prediction with the function:
# predict(obj, grid, BLUE=FALSE)
# this gives the prediction of Y(s_0):
predict(g.tr, s0.new)

## [using ordinary kriging]
##        coordinates var1.pred var1.var
## 1 (179180, 330100)  5.293158 0.1433444

# variance > 0 (as expected)

# Estimate the mean:use the argument 'BLUE'
predict(g.tr, s0.new, BLUE = TRUE)

## [generalized least squares trend estimation]
##        coordinates var1.pred var1.var
## 1 (179180, 330100)  6.053541 0.03981776

# this gives the estimate of the mean
# (trend component) under gls

## consider a location where I OBSERVE data
# this gives the prediction of Y(s_0)
# in the first location (zero variance!)
meuse[1,]
```

we're creating a gstat object containing all the informations we need to make predictions we need:
- the model we're using for the data (in this case stationary model)
- the data themselves
- the spatial dependence on the field

prediction (Z^*)

variance of the prediction

remember that it's the log!

we want the estimate of the mean in that location instead of the prediction in that location

BLUE = TRUE ⇒ instead of computing the best linear unbiased predictor (which is the kriging predictor) it provides the best linear unbiased estimator for the mean ($\hat{m}$)

estimate of the mean

variance of the estimation of the mean

!!!

≠ (very different!)

Here the prediction is different from the estimation of the mean (because our best point prediction takes advantage of the nearby locations)

In this case we can see that locations near our target tend to be lower than the mean → we use this information to have a lower prediction than mean

```
##        coordinates cadmium copper lead zinc  elev       dist   om ffreq soil
## 1 (181072, 333611)    11.7     85  299 1022 7.909 0.00135803 13.6     1    1
##   lime landuse dist.m
## 1    1      Ah     50
```

```
predict(g.tr, meuse[1,])
```

what happen when we try to make a prediction in a location in which we observed the data?
The prediction will be exactly the same as the data that we have!

```
## [using ordinary kriging]
```

```
##        coordinates var1.pred      var1.var
## 1 (181072, 333611) 6.929517 -1.110223e-16
```

this is the log ⇒ $e^{\wedge 6.929517} = 1022$

```
# this gives the estimate of the mean
# (drift component) under gls
predict(g.tr, meuse[1,], BLUE = TRUE)
```

what about the estimate of the mean of a zone in which we observe the data?
The mean will be ≠ since we're modeling the mean as constant in space.

```
## [generalized least squares trend estimation]
```

```
##        coordinates var1.pred   var1.var
## 1 (181072, 333611)  6.053541 0.03981776
```

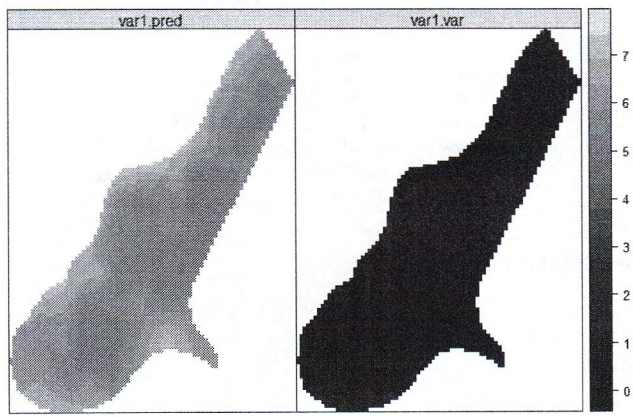
```
# prediction over the entire grid
lz.ok <- predict(g.tr, meuse.grid, BLUE = FALSE)
```

we can make predictions for all points of a grid (same function, but as an input we put a grid of locations)

```
## [using ordinary kriging]
```

```
spplot(lz.ok)
```

1.



spplot(lz.ok[,1])
spplot(lz.ok[,2])
if we want to see them separately

here we can check where we have more uncertainty (so if we have to decide where to put new points of sampling we can look at this)

```
### Non-stationary Univariate Spatial Prediction (Universal Kriging)
### ----------------------------------------------------------------
# the hypothesis of spatially constant mean may be too restrictive!
# we now use as covariate the square root of the distance from the river Meuse

# to fit the variogram on the residuals, one should take into account
# the spatial dependence while estimating the trend component by using GLS

# Create a gstat object setting a spherical (residual) variogram
# gstat(g.obj, id, formula, data, model, set,...)
meuse.gstat <- gstat(id = 'zinc', formula = log(zinc) ~ sqrt(dist),
                     data = meuse, nmax = 50, model=v.fit, set = list(gls=1))
meuse.gstat
```

(since we have evidence of anisotropy)

← to make the non stationary model we use the function gstat with a different formula:

log(zinc) ~ sqrt(dist)
this time we put a covariate

maximum number of iterations allowed for **GLS**

```
## data:
## zinc : formula = log(zinc)`~`sqrt(dist) ; data dim = 155 x 12 nmax = 50
## variograms:
##         model      psill    range
## zinc[1]   Nug 0.05065923   0.0000
## zinc[2]   Sph 0.59060463 896.9976
## set gls = 1;
```

model to start from in the fitting of the variogram: it should be a model from residuals but we're giving the model we created before. if we were using this function directly to make prediction this should be the model (fitted model) for residuals, the final one
(we'll recreate this object later on in order to put here the model estimated from the residuals)

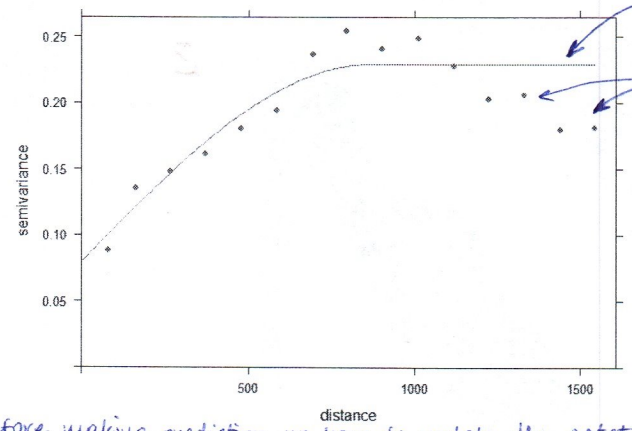
Once we have the object we estimate the variogram:

```
# Estimate the variogram from GLS residuals:
?variogram.gstat
```

```
## starting httpd help server ... done
```

```
v.gls<-variogram(meuse.gstat)
v.gls.fit <- fit.variogram(v.gls, vgm(1, "Sph", 800, 1))
plot(v.gls, v.gls.fit, pch = 19)
```

this is just to say that we want it to be estimated with GLS method



estimated model

variogram estimated from the residuals

Comparison with the other model: the nugget is more visible than before, so probably using a nugget is needed

Before making prediction we have to update the gstat model

```
# Update gstat object with variogram model
meuse.gstat <- gstat(id = 'zinc', formula = log(zinc) ~ sqrt(dist),
                     data = meuse, nmax = 50, model=v.gls.fit, set = list(gls=1))

## universal kriging:
## I have to define the covariate in s_0
s0.vec <- as.vector(slot(s0.new,'coords'))
# distance to the river: calculate the distance between s0 and each point of
# the river, then select the minimum
s0.dist <- min(rowSums(scale(meuse.riv,s0.vec)^2))
s0.new <- as.data.frame(c(s0.new,s0.dist))
names(s0.new) <- c('x','y','dist')
coordinates(s0.new) <- c('x','y')
s0.new <- as(s0.new, 'SpatialPointsDataFrame')
s0.new
```

this time we have to create the geographical location + the value of the regressor (distance from the river in this case)

```
##        coordinates     dist
## 1 (179180, 330100) 0.02006051
```

don't use covariance if we are not able to predict them in target locations !

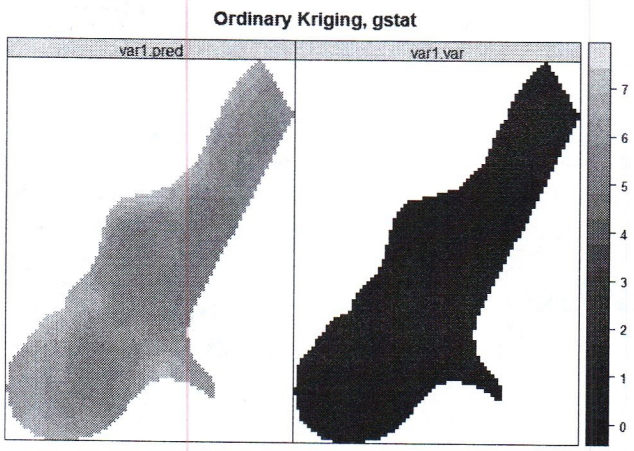
```
# Function "predict" uses the residual variogram stored in the gstat
# object to make the prediction
predict(meuse.gstat, s0.new)

## [using universal kriging]

##        coordinates zinc.pred  zinc.var
## 1 (179180, 330100)  6.493025 0.1761928
```

← prediction

```
# variance > 0 (as expected)

# this gives the estimate of x(s_0)'*beta
# (trend component) under gls
predict(meuse.gstat, s0.new, BLUE = TRUE)

## [generalized least squares trend estimation]

##        coordinates zinc.pred  zinc.var
## 1 (179180, 330100)  6.706735 0.04638581
```

← estimate of the drift at that location

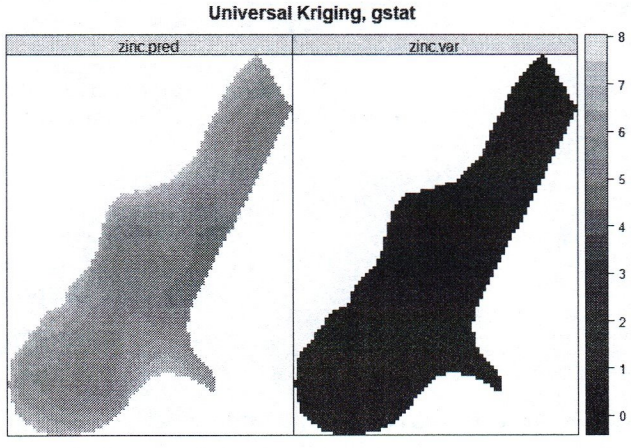
```
# prediction over the entire grid
lz.uk <- predict(meuse.gstat, meuse.grid, BLUE=FALSE)

## [using universal kriging]

# estimate of the mean over the entire grid
lz.uk.BLUE <- predict(meuse.gstat, meuse.grid, BLUE=TRUE)

## [generalized least squares trend estimation]

spplot(lz.ok, main = 'Ordinary Kriging, gstat')
```



(let's make a comparison between the 2 methods of kriging : after this)

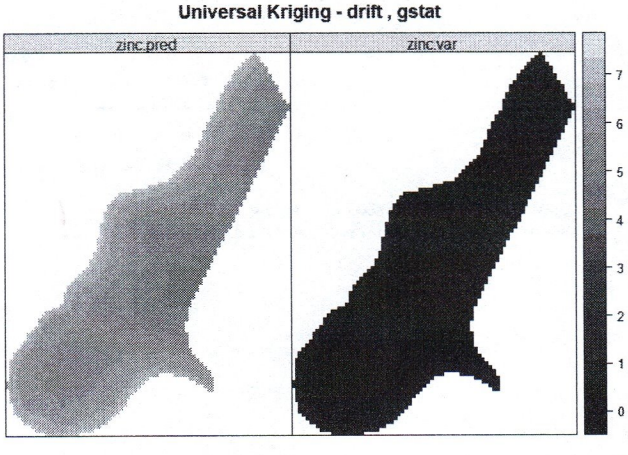
we've not coding it, but let's look at the pictures 1. and 2.

```
spplot(lz.uk, main = 'Universal Kriging, gstat')
```



2.

```
spplot(lz.uk.BLUE, main = 'Universal Kriging - drift , gstat')
```

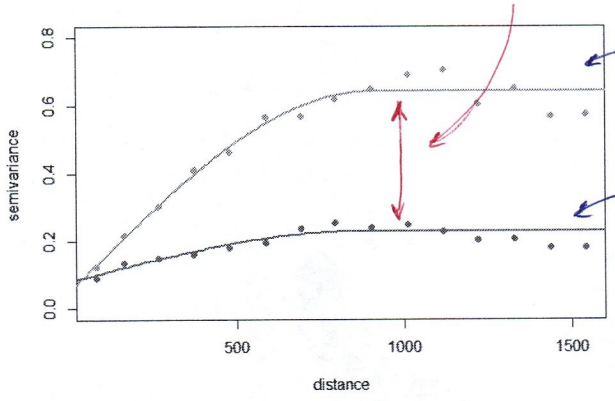


we can see that the drift is influencing a lot our final prediction

= we should use the stationary model or not?

```
# Is the drift important to explain the variability of the response variable?
# Let's compare the variogram of the data and of the residuals:
plot(v$dist,v$gamma,xlab='distance',ylab='semivariance',pch=19,col='skyblue1',ylim=c(0,0.8))
curve(v.f.est(x, C0=v.fit[2,2]+v.fit[1,2], cov.pars=rbind(c(v.fit[2,2], v.fit[2,3]),c(v.fit[1,2], v.fit[1,3])), cov.model =
 c("spherical","pure.nugget")), from = 0.0001, to = 1600,
      xlab = "distance", ylab = expression(gamma(h)),
      main = "Variogram model",add=TRUE,col='skyblue1',lwd=2, ylim=c(0,110))
points(v.gls$dist,v.gls$gamma,xlab='distance',ylab='semivariance',pch=19,col='steelblue',ylim=c(0,0.8))
curve(v.f.est(x, C0=v.gls.fit[2,2]+v.gls.fit[1,2],
              cov.pars=rbind(c(v.gls.fit[2,2], v.gls.fit[2,3]),c(v.gls.fit[1,2], v.gls.fit[1,3])), cov.model = c("spherical"
,"pure.nugget")), from = 0.0001, to = 1600,
      xlab = "distance", ylab = expression(gamma(h)),
      main = "Variogram model",add=TRUE,col='steelblue',lwd=2, ylim=c(0,110))
```

amount of variability explained by the data



variogram estimated from the data

variogram estimated from the residuals

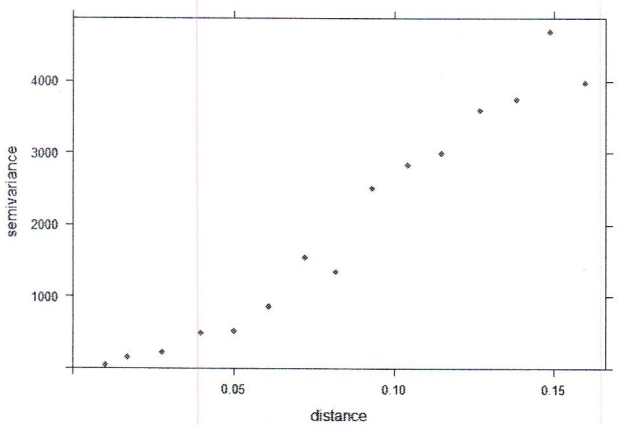
The difference (in amplitude) between the 2 is given by the drift term (⇒ when we're modeling a drift term we'll always have a lower variance of the process because we're explaining part of the variability of the process with the drift term)

```
### ------------------------------------------------------------------------
###
### EXERCIZES FROM PAST EXAMS
###
### ------------------------------------------------------------------------
```

```
###-------------------------------------------------------------------------
### Exercise 1
###-------------------------------------------------------------------------
# One of the most relevant consequences of the eruption of volcan
# Eyjafjoll (in Iceland), in 2010, is the contamination by fluoride.
# The latter is due to the deposit on the ground of the ash released
# in the atmosphere during the eruption.
# The file "fluoruro.txt" reports the coordinates of 50 measurement sites
# s_i, i=1,...,50, the corresponding concentrations of fluoride (ppm) F(s_i)
# and the distance D.s_i of each site s_i from the crater of the volcano.
# Denoting by delta a zero-mean, second-order stationary and isotropic random
# field:
# a) Estimate two empirical variograms, assuming the following models:
#    F(s_i)=beta_0+delta(s_i) and
#    F(s_i)=beta_0+beta_1*D.s_i+delta(s_i).
#    Choose the most appropriate model for the observations.
# b) Fit to the empirical variogram at point (a), a Gaussian model
#    without nugget, via weighted least squares. Use as initial parameters:
#    sill=100, range=0.02. Report the estimates of sill and range.
# c) Fit to the empirical variogram chosen at point (a), a spherical model
#    without nugget, via weighted least squares. Report the estimates of sill
#    and range.
# d) Compare the variograms estimated at points (b) and (c), with the empirical
#    variogram at point (a). Given that the ash deposition is known to be
#    a very regular phenomenon, which variogram model is the most appropriate?
# e) Based on model (d), estimate the concentration of fluoride due to the eruption
#    in the city of Raufarhofn (s0 = (0.3; 0.24), D.s0 = 0.1970)
# f) Based on model (d), estimate the concentration of fluoride at the same
#    location, due to a possible new eruption of equivalent intensity, independent
#    of that of 2010.

###-------------------------------------------------------------------------
### SOLUTION
###-------------------------------------------------------------------------
# Data import
data=read.table('fluoruro.txt')
names(data)[3]='f'
attach(data)
coordinates(data)=c('X','Y')

# a) Estimate two empirical variograms, assuming the following models:
#    F(s_i)=beta_0+delta(s_i) and
#    F(s_i)=beta_0+beta_1*D.s_i+delta(s_i).
#    Choose the most appropriate model for the observations.

v=variogram(f ~ 1, data=data)
plot(v,pch=19)
```

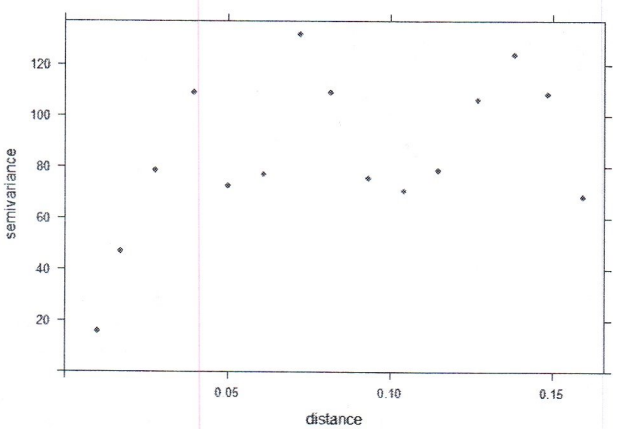
X	Y	concentration	distance

stationary model
non-stationary model



← this doesn't seem to stabilize to an asymptote

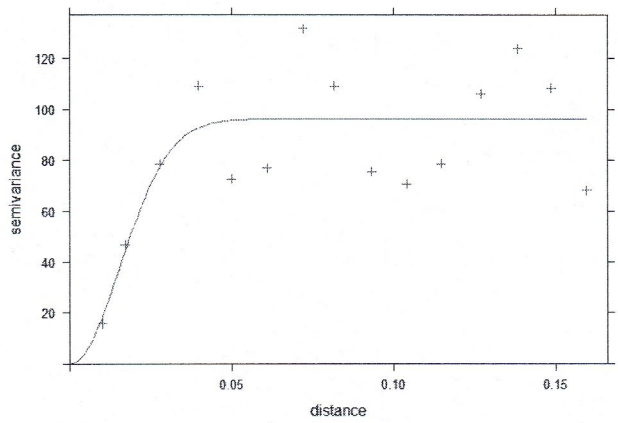
```
v.t=variogram(f ~ D, data=data)
plot(v.t,pch=19)
```



← for sure now it's stabilizing on an asymptote

(Between the two models we should choose the 2nd)

```
# b) Fit to the empirical variogram at point (a), a Gaussian model
#    without nugget, via weighted least squares. Use as initial parameters:
#    sill=100, range=0.02. Report the estimates of sill and range.
v.fit2 <- fit.variogram(v.t, vgm(100, "Gau", 0.02))
plot(v.t, v.fit2, pch = 3)
```

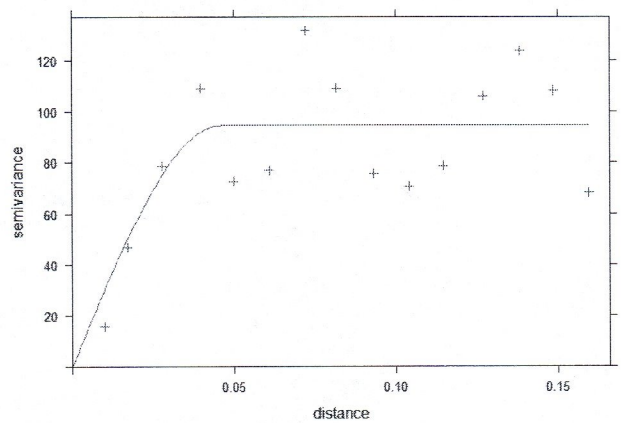



```
v.fit2
```

```
##   model    psill      range
## 1   Gau 96.03304 0.0217514
```

this sometimes is just a value proportional at the range (if we see the range we'll notice that it's not 0.02)

```
# c) Fit to the empirical variogram chosen at point (a), a spherical model
#    without nugget, via weighted least squares. Report the estimates of sill
#    and range.
v.fit1 <- fit.variogram(v.t, vgm(100, "Sph", 0.08))
plot(v.t, v.fit1, pch = 3)
```




```
v.fit1
```

```
##   model    psill      range
## 1   Sph 94.38304 0.04616408
```

```
# d) Compare the variograms estimated at points (b) and (c), with the empirical
#    variogram at point (a). Given that the ash deposition is known to be
#    a very regular phenomenon, which variogram model is the most appropriate?

# We choose v.fit2 ⟶ gaussian

# e) Based on model (d), estimate the concentration of fluoride due to the eruption
#    in the city of Raufarhofn (s0 = (0.3; 0.24), D.s0 = 0.1970)

g.t <- gstat(formula = f ~ D, data = data, model = v.fit2)

D.s0=0.1970
s0=as.data.frame(matrix(c(0.3,0.24,D.s0),1,3))
names(s0)=c('X','Y','D')
coordinates(s0)=c('X','Y')

predict(g.t, s0, BLOCK = FALSE)
```

very regular = very smooth

```
## [using universal kriging]
```

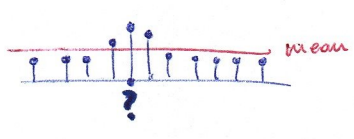
```
##   coordinates var1.pred var1.var
## 1 (0.3, 0.24)  50.80604 98.66119
```

```
# f) Based on model (d), estimate the concentration of fluoride at the same
#    location, due to a possible new eruption of equivalent intensity, independent
#    of that of 2010.

predict(g.t, s0, BLOCK = TRUE)
```

Now we're talking about a new realization of the field which is ⊥ of the previous one ⟹ the new realizations at the target will be ⊥
⟹ to make prediction in the independent case we can just use the mean

```
## [generalized least squares trend estimation]
```

```
##   coordinates var1.pred var1.var
## 1 (0.3, 0.24)  50.66903 3.077244
```



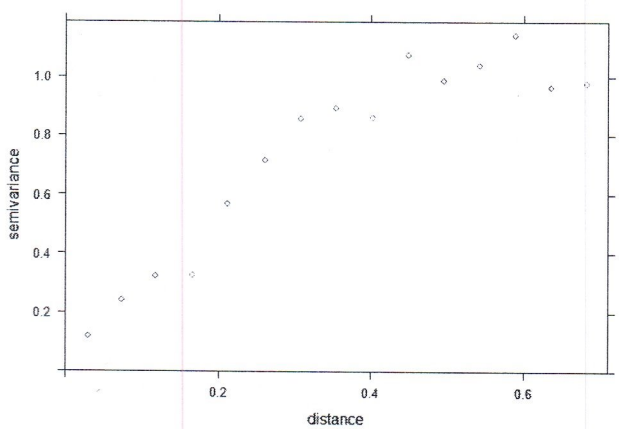
PREDICTION (SPACE):
we're taking into account that the nearby zones of the target location have a value > mean
⟹ the prediction will be higher than the mean

PREDICTION IN TIME:
we don't know how it will be; so the only thing that we can say it's the mean:

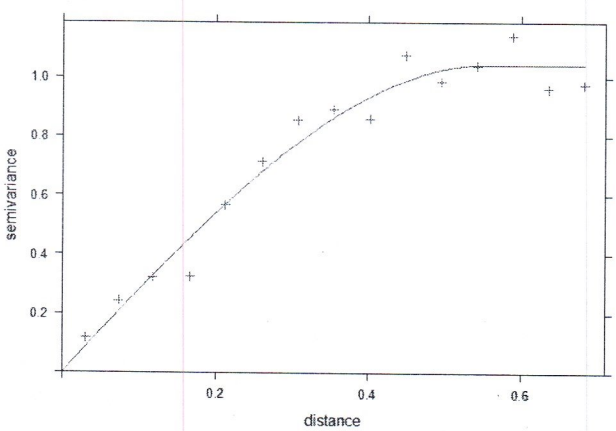
```
### ------------------------------------------------------------------------
### Exercise 2
### ------------------------------------------------------------------------
# The file radioville.txt reports the information on 158 control units
# in the area around the nuclear power plant of Radioville.
# At each site, available data consist of: radioactivity levels [Bq],
# longitude [Â°N], latitude [Â°W] and type of soil [urban/vegetation].
# Denoting by s_i the i-th site, by R the radioactivity level,
# by eps a weakly stationary random field and by D a dummy
# urban/vegetation:

# a) estimate the parameters of the linear model
#    R(s_i) = beta_0 + beta_1 D(s_i) + eps(s_i)
#    assuming for eps a spherical variogram without nugget, estimated
#    via weighted least squares;
# b) estimate the parameters of the linear model
#    R(s_i) = beta_0 + beta_1 D(s_i) + eps(s_i)
#    assuming for eps a spherical variogram with nugget, estimated
#    via weighted least squares;
# c) choose the best variogram model by comparing the fitted model
#    with the corresponding empirical variograms (report qualitative
#    plots and the estimated variogram parameters)
# d) on the basis of model (c), predict the radioactivity level at the
#    parking lot of the shopping centre of Radioville (lon = 78.59,
#    lat = 35.34), and in the park of Radioville (lon = 77.6,
#    lat = 34.99);
# e) estimate variance of prediction error at the same locations
#    as at point d).

### ------------------------------------------------------------------------
### SOLUTION
### ------------------------------------------------------------------------

data <- read.table('radioville.txt',header=TRUE)
attach(data)
```

```
## The following object is masked from data (pos = 3):
##
##     D
```

```
# create dummy: 0 = urban, 1 = vegetation
DUMMY <- rep(0,length(D))
DUMMY[which(D=='V')] <- 1
data <- data.frame(cbind(Bq,Long,Lat,DUMMY))
names(data) <- c('Bq','Long','Lat','D')
coordinates(data) <- c('Long','Lat')

## point a)
## fitting a variogram without nugget

v <- variogram(Bq ~ D, data = data)
plot(v)
```




```
v.fit1 <- fit.variogram(v, vgm(1, "Sph", 0.5))
plot(v, v.fit1, pch = 3)
```




```
v.fit1
```

```
##   model    psill     range
## 1   Sph 1.042401 0.5549304
```

```
# coefficient of the linear model:
# it sufficies to estimate the drift at two locations where we have observations,
# with D=U and D=V
# data[1,] = urbane
# data[6,] = vegetation
g.tr <- gstat(formula = Bq ~ D, data = data, model = v.fit1)
predict(g.tr, data[1,], BLUE = TRUE)
```

```
## [generalized least squares trend estimation]
```

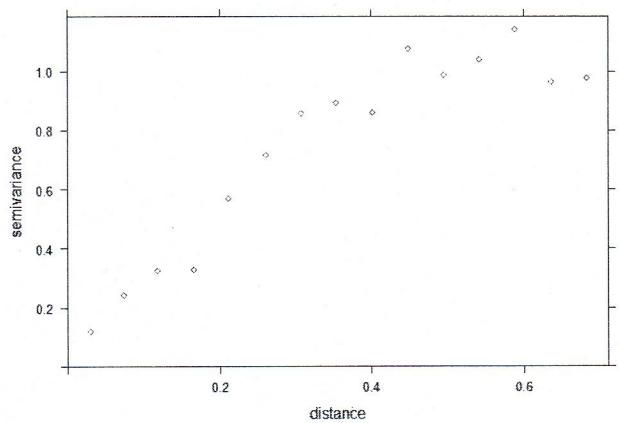
```
##        coordinates var1.pred  var1.var
## 1 (78.614, 34.916)  3.988671 0.0644284
```

```
predict(g.tr, data[6,], BLUE = TRUE)
```

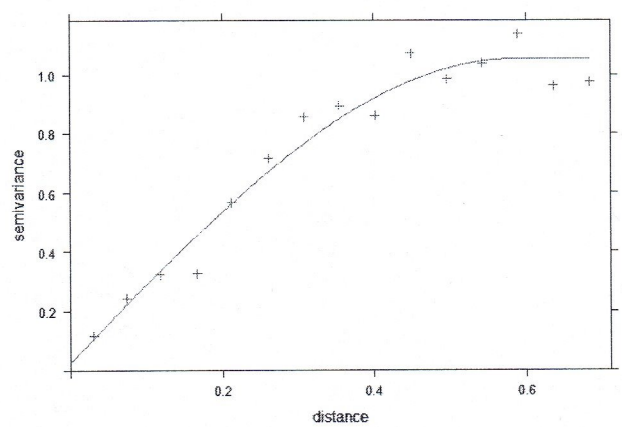
```
## [generalized least squares trend estimation]
```

```
##        coordinates var1.pred var1.var
## 1 (78.576, 35.597)  7.932618 0.101812
```

```
## point b)
## fitting a variogram with nugget

v <- variogram(Bq ~ D, data = data)
plot(v)
```




```
v.fit2 <- fit.variogram(v, vgm(0.6, "Sph", 0.5, 0.1))
plot(v, v.fit2, pch = 3)
```




```
v.fit2
```

```
##   model      psill    range
## 1   Nug 0.02782848 0.000000
## 2   Sph 1.02569913 0.583282
```

```
# it sufficies to estimate the drift at two locations where we have observations,
# with D=U and D=V
# data[1,] = urbane
# data[6,] = vegetation
g.tr <- gstat(formula = Bq ~ D, data = data, model = v.fit2)
predict(g.tr, data[1,], BLUE = TRUE)
```

```
## [generalized least squares trend estimation]
```

```
##        coordinates var1.pred   var1.var
## 1 (78.614, 34.916)  3.985999 0.06875351
```

```
predict(g.tr, data[6,], BLUE = TRUE)
```

```
## [generalized least squares trend estimation]
```

```
##         coordinates var1.pred  var1.var
## 1 (78.576, 35.597)  7.923632 0.1131944
```

```
## point d)
## predict at 2 new locations: we use model 1 (without nugget)
g.tr <- gstat(formula = Bq ~ D, data = data, model = v.fit1)

# urbane : 78.59,35.34
s0.new <- as.data.frame(matrix(c(78.59,35.34,0),1,3))
names(s0.new) <- c('lon','lat','D')
coordinates(s0.new) <- c('lon','lat')
predict(g.tr, s0.new)
```

```
## [using universal kriging]
```

```
##       coordinates var1.pred  var1.var
## 1 (78.59, 35.34)  5.459977 0.1702403
```

```
# vegetation : 77.69,34.99
s0.new <- as.data.frame(matrix(c(77.69,34.99,1),1,3))
names(s0.new) <- c('lon','lat','D')
coordinates(s0.new) <- c('lon','lat')
predict(g.tr, s0.new)
```

```
## [using universal kriging]
```

```
##       coordinates var1.pred  var1.var
## 1 (77.69, 34.99)  7.721414 0.6281572
```