Geostatistics

Alexander J Ohrt, Mikel, Victor (add apellidos and order alphabetically)

5 november, 2021

This is the first assignment in Spatial Epidemiology at UPC, fall 2021. It is in the topic of Geostatistics. Don't know if we should write anything else in this "intro".

Load libraries and data

```
# The libraries we used.
library(geoR)
library(sm)
library(gstat)

coordinates <- read.table("poly84.txt", head = TRUE, sep = "\t", dec = ".")
elevs <- read.table("elevationsIslet.txt", head = TRUE, sep = "\t", dec = ".")
head(elevs)
```

```
#>           x          y data
#> 1  98.57754  0.8906123 17.6
#> 2 113.59737 19.5085930 27.1
#> 3 110.53511 39.6851815 16.4
#> 4 105.27965 28.3568305 -9.5
#> 5  95.20204  8.4319902  6.1
#> 6 102.96788 53.0629008 -0.5
```

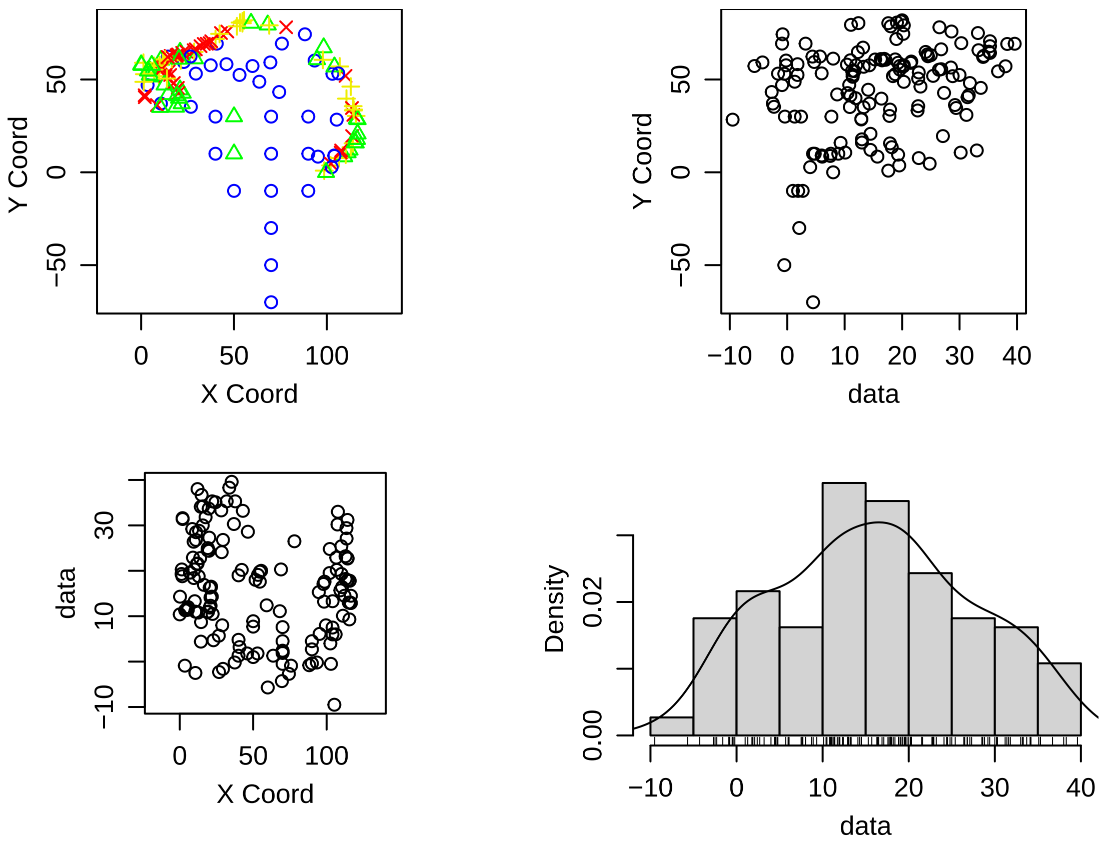
1. Exploration of the large scale variability of the elevation

In this problem we will explore the large scale variability of the elevation data.

```
geoelevs <- as.geodata(elevs)
summary(geoelevs)
```

```
#> Number of data points: 148
#>
#> Coordinates summary
#>            x         y
#> min   0.0000 -70.00000
#> max 116.5471  81.88936
#>
#> Distance summary
#>         min         max
#>   0.2104305 152.5866269
#>
#> Data summary
#>     Min.   1st Qu.   Median     Mean  3rd Qu.     Max.
#> -9.50000  7.67500 16.35000 15.97095 24.17500 39.60000
```

```
plot(geoelevs)
```

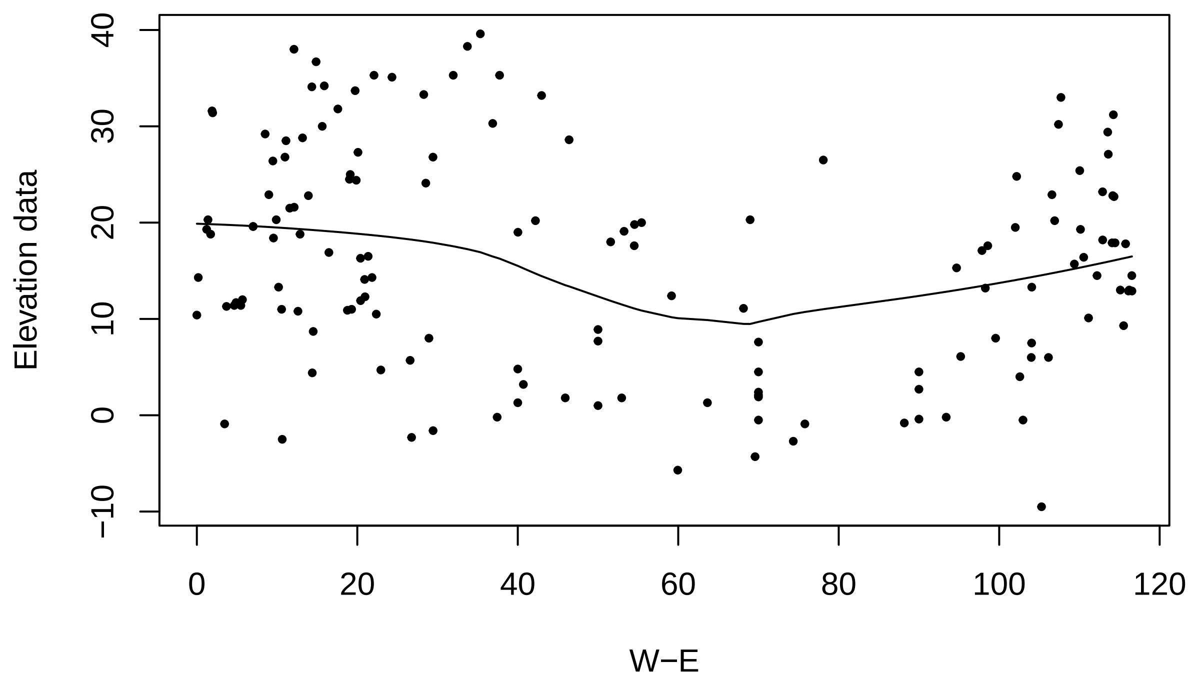


```
# Remember that:
###   (circles) : 1st quartile
###   (triangles) : 2nd quartile
###   (plus)  : 3rd quartile
###   (crosses)  : 4th quartile
```

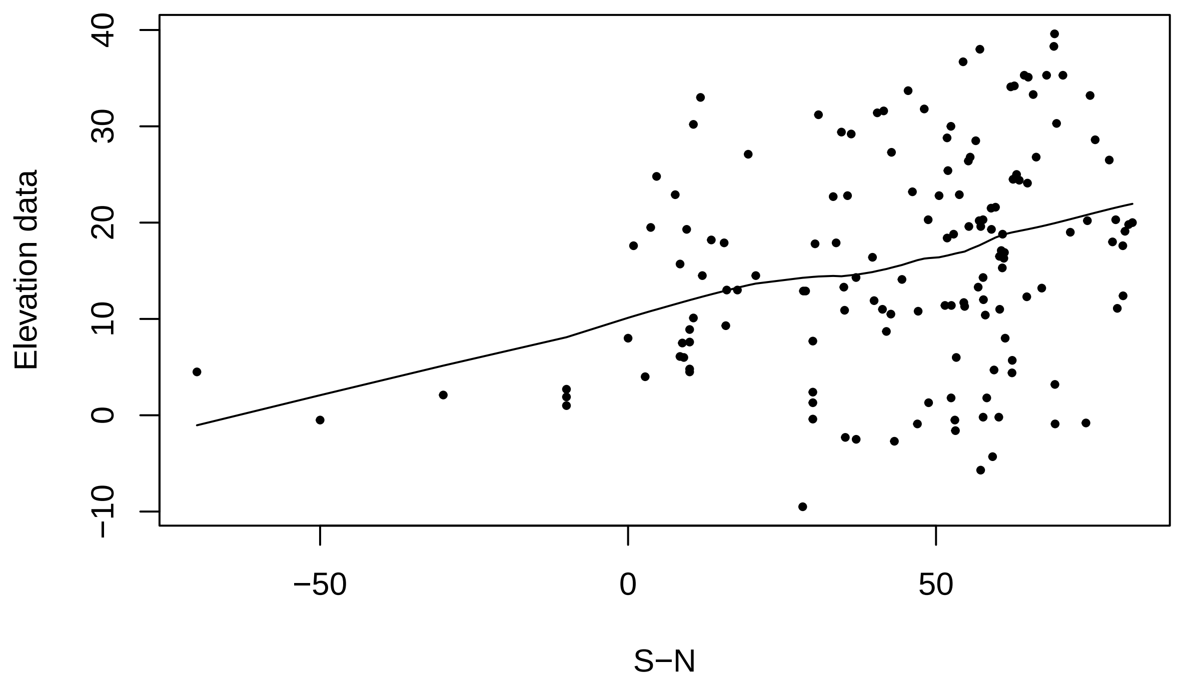
The histogram of the elevations shows that the data is relatively symmetrically distributed, resembling a normal distribution, which means that we will not need to apply a transformation at this stage. Moreover, the plot in the upper left corner shows that most of the larger values are in the upper left part of the islet, based on the red crosses, which make up the largest 25% of values in the data set. In general, the first quartile in the data is more widespread on the islet, while the rest of the data is mostly spread out around the border of the islet. Note that the plot in the lower left shows a trend in the longitude, perhaps following a second order polynomial (upside down U). There are also some indications of a rising trend in the latitude, based on the plot in the upper right corner. An attempt at removing these trends will be done later, using a linear model.

The following plots further ground the initial theories about the trends in the latitude and longitude.

```
with(geoelevs, plot(coords[, 1], data, xlab = "W-E",
              ylab = "Elevation data", pch = 20, cex = 0.7))
lines(with(geoelevs, lowess(data ~ coords[, 1])))
```



```
with(geoelevs, plot(coords[, 2], data, xlab = "S-N",
                    ylab = "Elevation data", pch = 20, cex = 0.7))
lines(with(geoelevs, lowess(data ~ coords[, 2])))
```



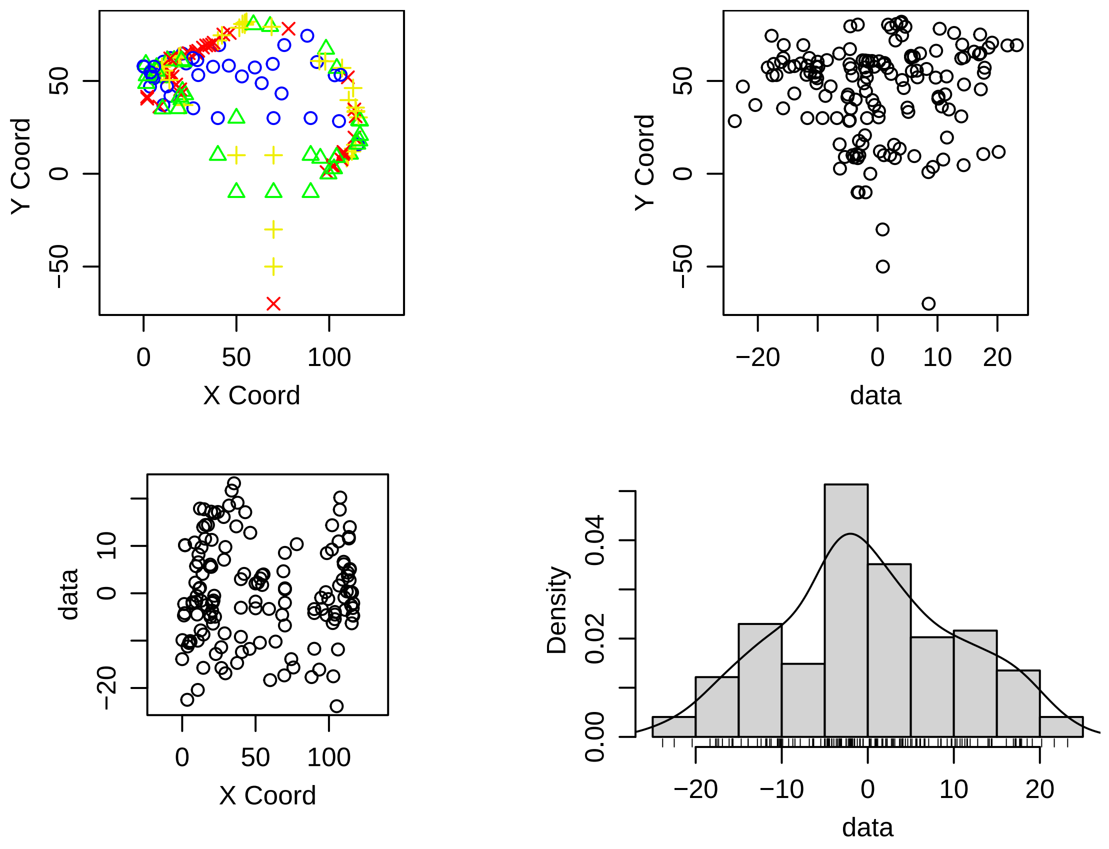
Attempting to remove the trends with a linear model. The residuals of the linear regression are added to a new dataframe, with the original data.

```
lm.fit <- lm(data ~ y + poly(x,2), data = elevs)
summary(lm.fit)
```

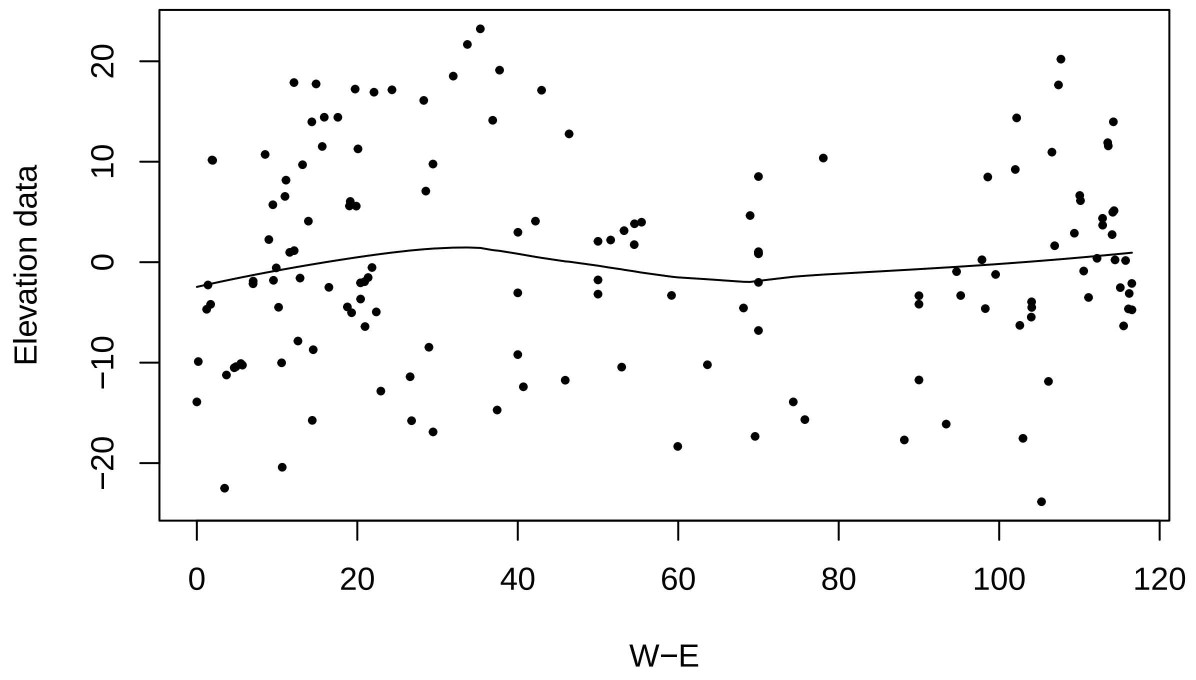
```
#>
#> Call:
#> lm(formula = data ~ y + poly(x, 2), data = elevs)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -23.845  -6.301  -1.079   6.571  23.226
#>
#> Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 10.18780    1.81117   5.625 9.35e-08 ***
#> y            0.13223    0.03658   3.615 0.000415 ***
#> poly(x, 2)1 -7.12030   11.65722  -0.611 0.542291
#> poly(x, 2)2 39.06998   10.41746   3.750 0.000255 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 10.33 on 144 degrees of freedom
#> Multiple R-squared:  0.1754, Adjusted R-squared:  0.1583
#> F-statistic: 10.21 on 3 and 144 DF,  p-value: 3.862e-06
```

```
# Add residuals to a new data frame.
elevs2 <- data.frame(elevs, residuals = lm.fit$residuals)
geoelevs2 <- as.geodata(elevs2, data.col = 4)
```

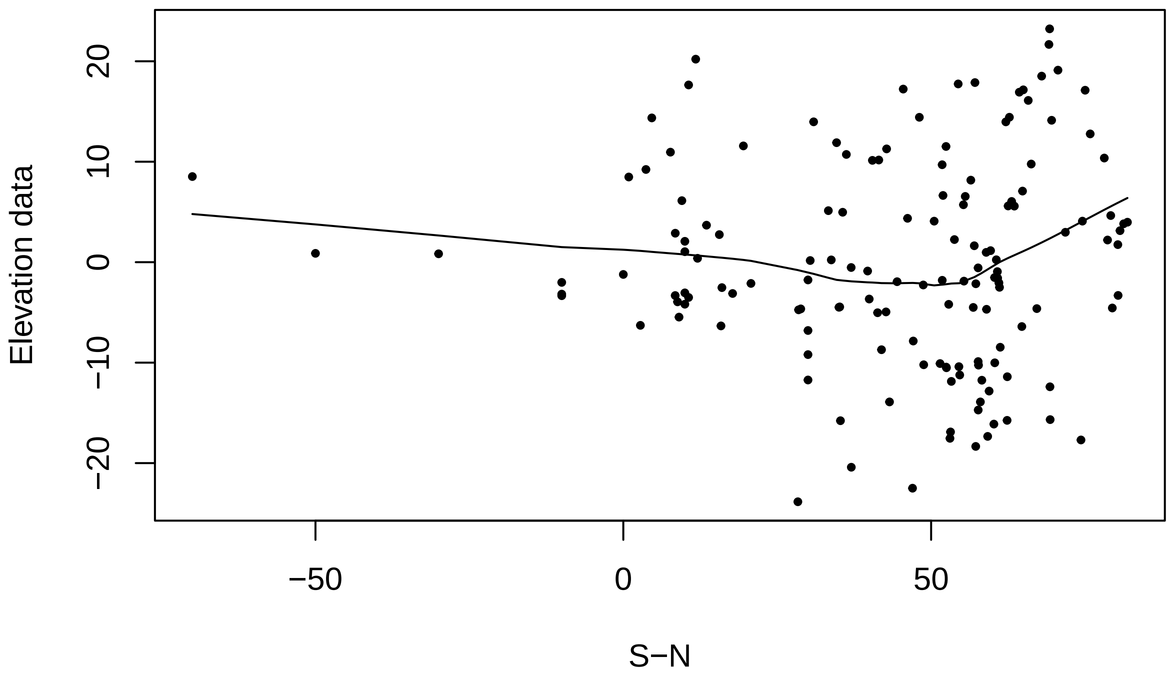
```
plot(geoelevs2)
```



```
with(geoelevs2, plot(coords[, 1], data, xlab = "W-E",
                     ylab = "Elevation data", pch = 20, cex = 0.7))
lines(with(geoelevs2, lowess(data ~ coords[, 1])))
```



```
with(geoelevs2, plot(coords[, 2], data, xlab = "S-N",
                     ylab = "Elevation data", pch = 20, cex = 0.7))
lines(with(geoelevs2, lowess(data ~ coords[, 2])))
```



It looks like some of the trend has been removed, i.e. the linear model has explained some of the variations in the data. Note that we also attempted different combinations of dropping the second order term in longitude and adding a second order term in latitude, but chose the above model for its simplicity and reasonably good performance. It is the simplest model among those we tried, where all the models had very similar residuals. Note that this is by no means a rigorous analysis when applying the linear model, but we have chosen the simplest model that seems to remove some of the trend in the data. Therefore, we will use the residuals from the linear regression in the rest of the analysis. Ok?

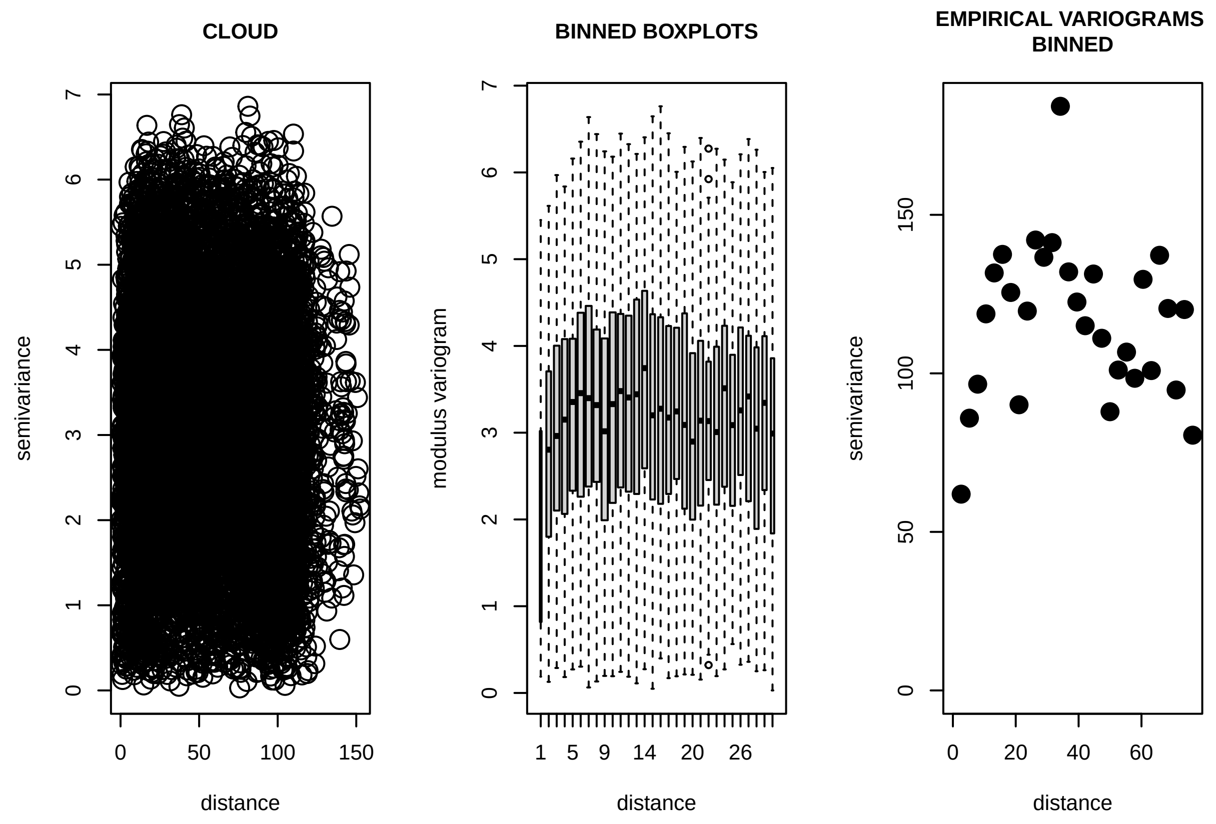
2. Exploration of the small scale variability of the elevation

In this problem we will explore the small scale variability of the elevation data.

First we calculate and plot the variogram cloud together with the empirical variogram. Remember that we now are analyzing the residuals from the linear regression presented earlier. Note that the robust estimator (`modulus`, Hawkins and Cressie 1993) is used when estimating the empirical variogram.

```
# Practical rules: lags only up to half of maxdist. That is why maxdist is saved here.
maxdist <- max(dist(cbind(elevs$x,elevs$y)))
cloud <- variog(geoelevs2, option = "cloud", estimator.type = "modulus")
bp <- variog(geoelevs2, option = "bin", bin.cloud = T,
               pairs.min=30, max.dist=maxdist/2,
               estimator.type = "modulus", uvec=seq(0,maxdist/2,l=30))
bin <- variog(geoelevs2, option = "bin", pairs.min=30, max.dist=maxdist/2,
                estimator.type = "modulus", uvec=seq(0,maxdist/2,l=30))
# Also: pairs,min = 30 since the sample variogram should only be considered for lags
# that have more than 30 pairs.
```

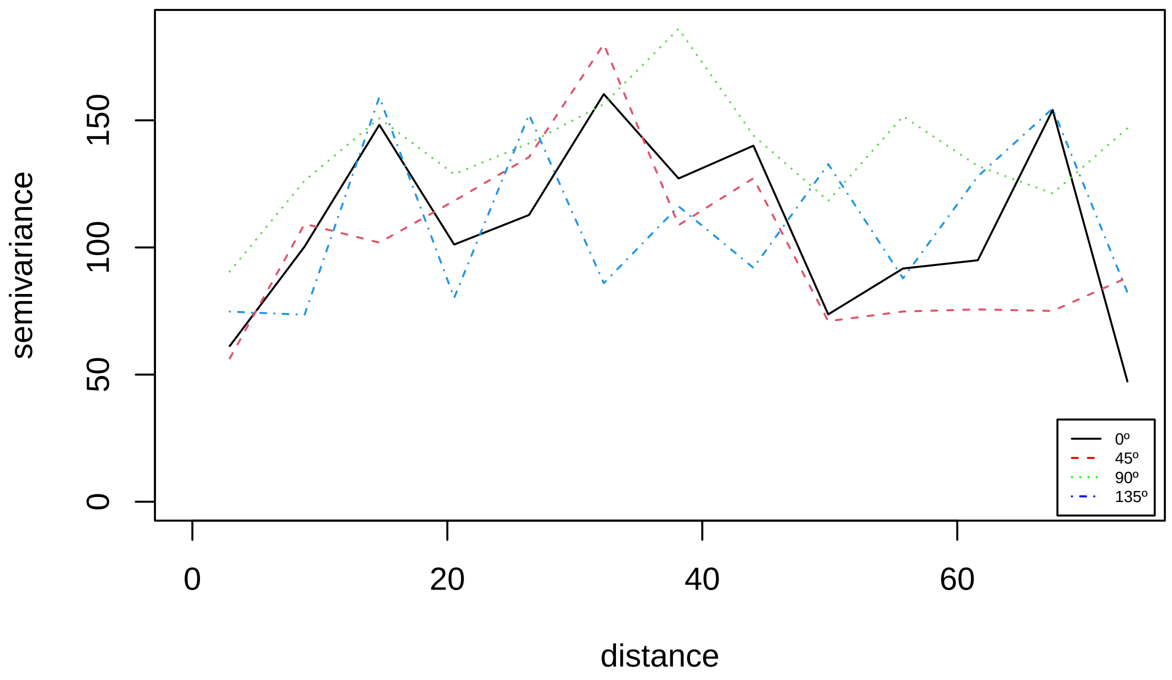
```r
par(mfrow=c(1,3))
plot(cloud,main="CLOUD", cex.main=1, cex.lab=1, cex=2)
plot(bp, bin.cloud=T, cex.lab=1)
title(main = "BINNED BOXPLOTS", cex.main=1)
plot(bin, main="EMPIRICAL VARIOGRAMS \nBINNED",cex.main=1, cex.lab=1, cex=2, pch=16)
```



From these variogram plots it looks like the range of a variogram is approx. 20 (distance), the sill around 140 (semivariance) and the nugget might be around 30. Thus, one might say that there is spatial correlation, at least in a range of 30. The bin at approximately distance 35 is an outlier and will be treated as such in the following. However, we will not remove it from bin.

Will calculate directional variograms to study isotropic/anistropic properties also. Kept this here instead of in problem3, but it could be moved there also.

```r
par(mfrow=c(1, 1))
variod <- variog4(geoelevs2,max.dist=maxdist/2, pairs.min=30,estimator.type = "modulus")
plot(variod,lyt=2,legend=FALSE)
legend(x="bottomright", inset=0.01, lty=c(1,2,3,4), col=c("black", "red", "green","blue"),
       legend=c("0º", "45º", "90º","135º"), cex=0.5)
```

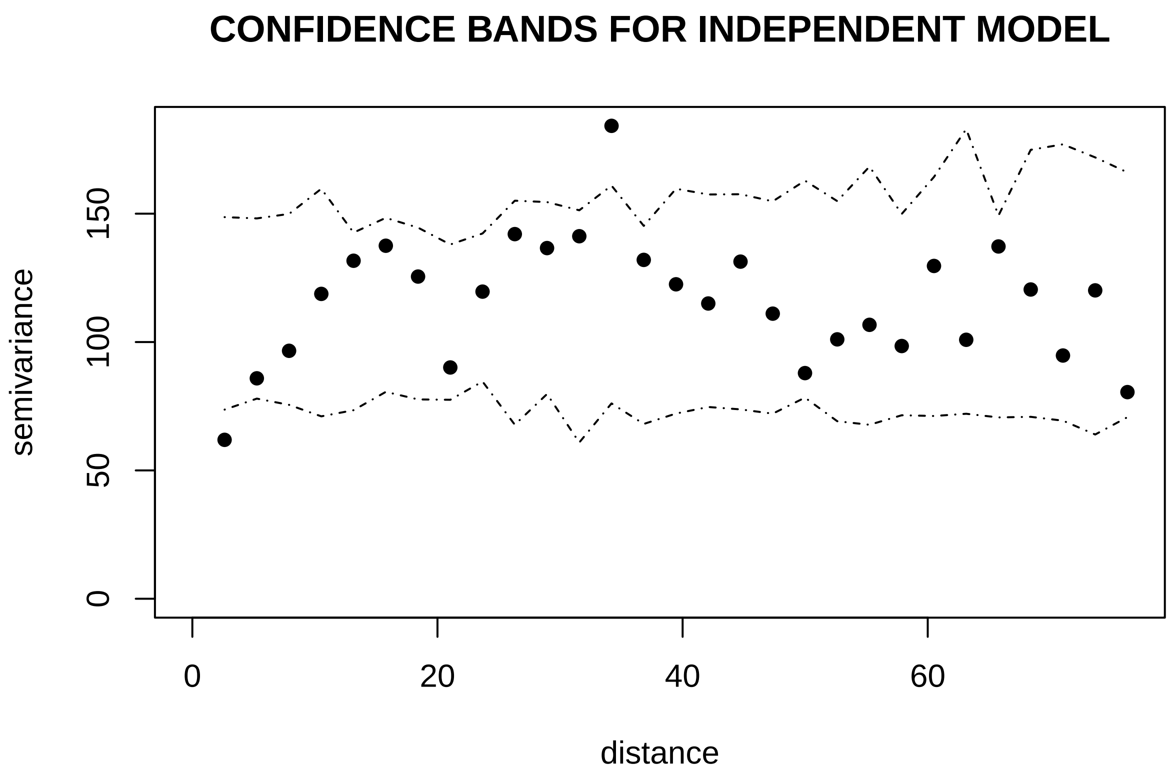


At first glance the data looks relatively isotropic, at least based on the 4 directions plotted above, which look relatively similar to each other.

3. Exploration of the spatial independence

In this problem we will further explore the spatial independence of the process. As specified in the problem description, we will set the seed to 1000.

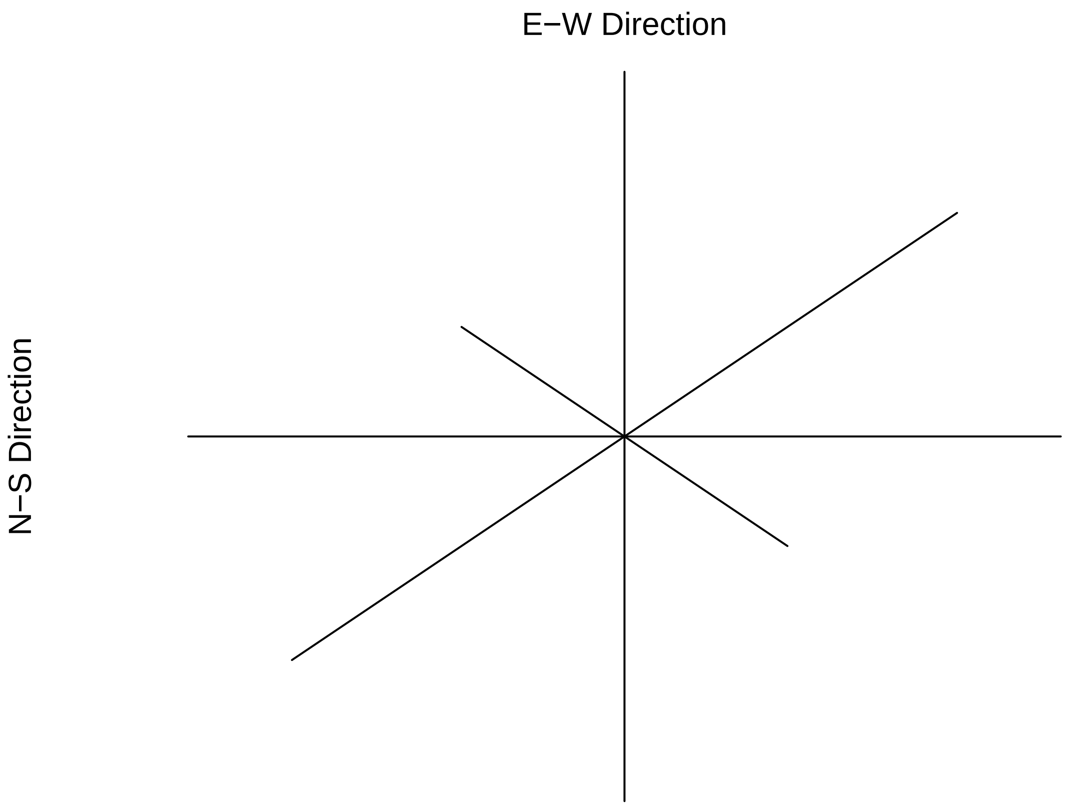
```
set.seed(12345)
par(mfrow=c(1,1))
indep.env <- variog.mc.env(geoelevs2, coords=geoelevs2$coords,
                        data=geoelevs2$data, obj.variog=bp, nsim=200)
plot(bp, envelope = indep.env, main="CONFIDENCE BANDS FOR INDEPENDENT MODEL",
                        lwd=2, pch=16)
```



The confidence bands show a range of estimated variograms when the measurements are randomly allocated to the spatial points. The lower confidence band is the minimum value of the variogram for the simulated data in each lag and, similarly, the upper confidence band is the maximum value. The hypothesis that the process is independent may be rejected if the empirical variogram falls outside these confidence bands, since this would indicate that it is unplausible that the variaogram has those values by chance. The plot shows that two of the points are outside the confidence bands (including the large outlier), which could indicate that complete spatial randomness in the underlying process may be unplausible. However, disregarding the outlier we think this plot is not enough to disregard the hypothesis of spatial randomness, since the rest of the points are inside the confidence bands. Thus we think the evidence against the null hypothesis is not strong enough and conclude that the data are spatially independent. This depends on the seed used to generate pseudorandom numbers, but it seems like the conclusion holds with other seeds also.

We will also add a Rose diagram, to study the anisotropy further. The Rose diagram will be plotted used the code supplied in the lectures (`RoseDiagram.R`).

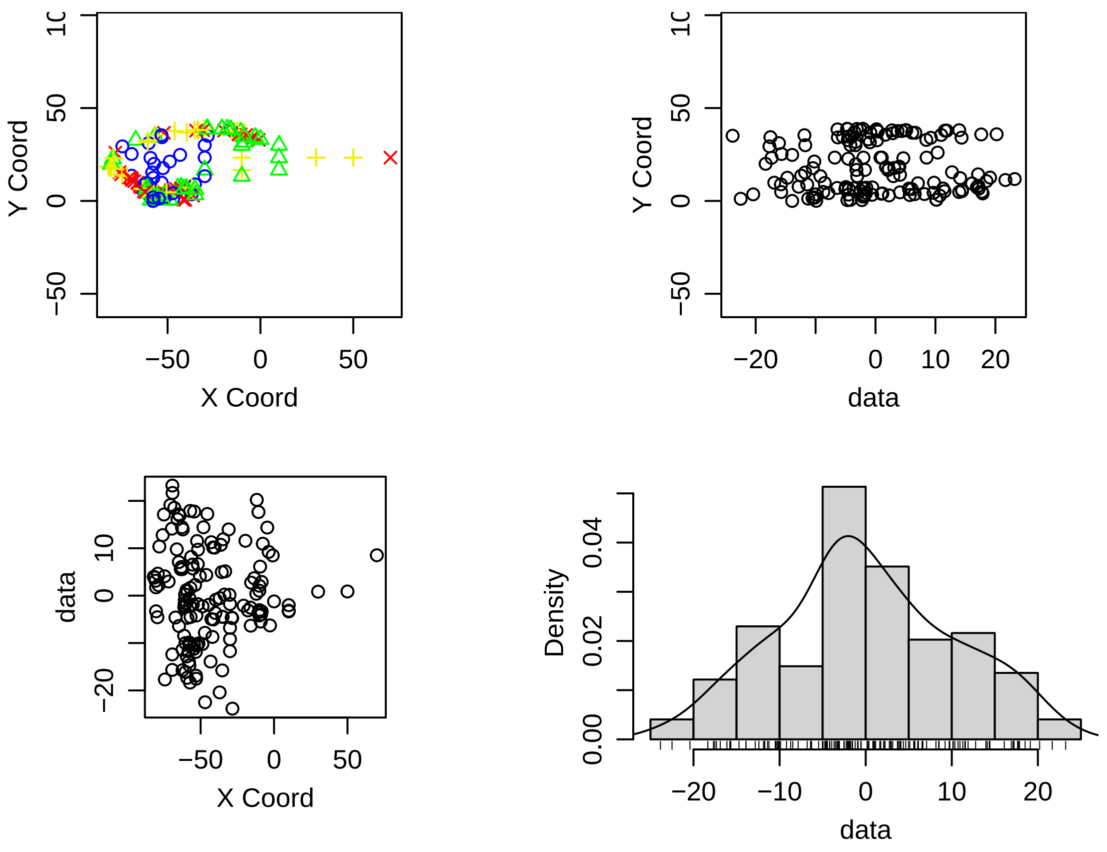
```
# Added a large crit.val since it has no default in the rose.diagram function. If it is larger than min
# it is automatically set to min(max.semi)*0.99 inside the function.
rose.diagram(data.var=geoelevs2$data,data.cds=geoelevs2$coord,max.dist=maxdist/2,
             numcases=15,numdirec=4,poly.tnd="cte", crit.val = 1000)
```



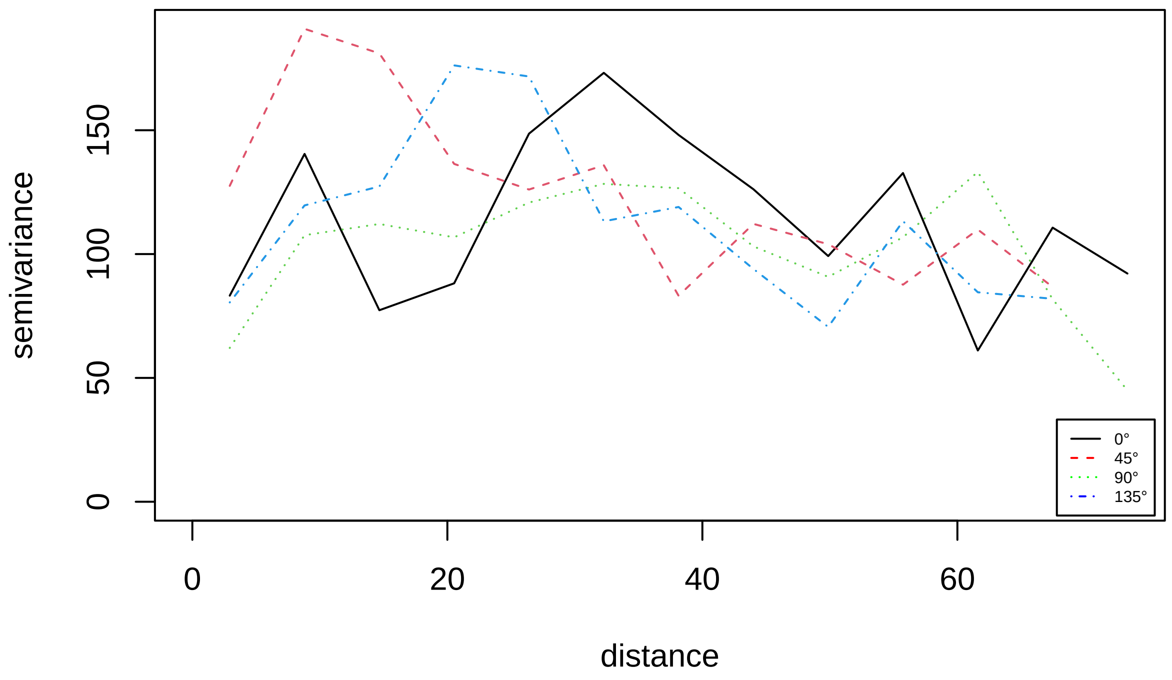
The idea behind this plot is that one can see if the data looks anisotropic. If this is the case, one can try to apply an anisotropic coordinate correction (rotational transformation). After the transformation, one can make the same plot as above again to check if the data looks more isotropic.

From the Rose Diagram plotted above, it looks like the Anisotropy angle is $\frac{\pi}{2}$ (angle between y-axis and the direction with the maximum range). Moreover, it looks like the Anisotropy ratio is approximately 3 (ratio between maximum and minimum ranges). Thus we can try the rotational transformation below and observe if it looks more isotropic.

```r
angle <- pi/2
ratio <- 3
elevs2.ani <- cbind(coords.aniso(geoelevs2$coords, c(angle,ratio),reverse = FALSE),geoelevs2$data)
geoelevs2.ani <- as.geodata(elevs2.ani)
plot(geoelevs2.ani)
```



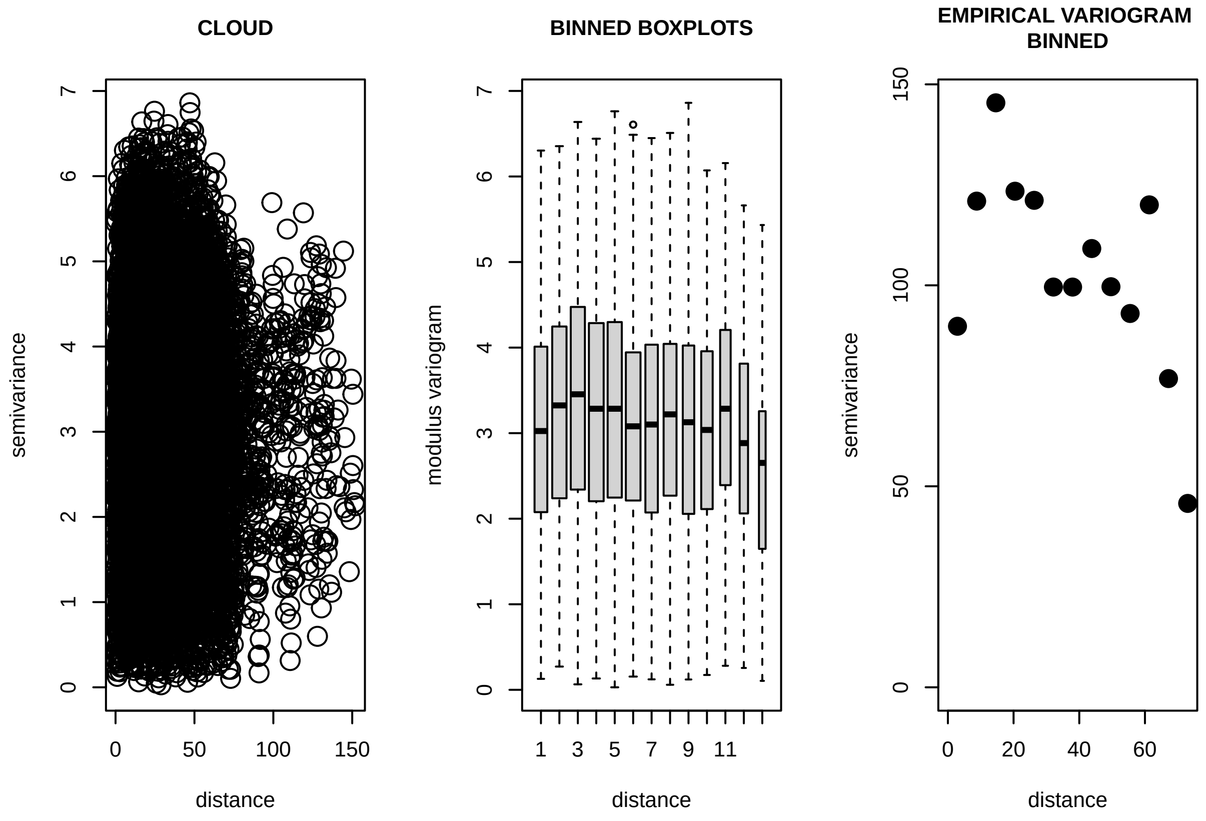
```
par(mfrow=c(1,1))
variod.ani <- variog4(geoelevs2.ani, max.dist=maxdist/2, pairs.min=30,estimator.type = "modulus");
plot(variod.ani,lyt=2,legend=FALSE)
legend(x="bottomright", inset=0.01, lty=c(1,2,3,4), col=c("black", "red", "green","blue"),
       legend=c("0\u00B0", "45\u00B0", "90\u00B0","135\u00B0"), cex=0.5)
```



Comparing the plot above to the same plot from earlier we conclude that this process looks less isotropic than before. Therefore, we will keep analyzing the data as it is. In this way we also avoid the need of transforming the coordinate data as well. What do you think about this conclusion? And I say just remove the code/plots from below as well.

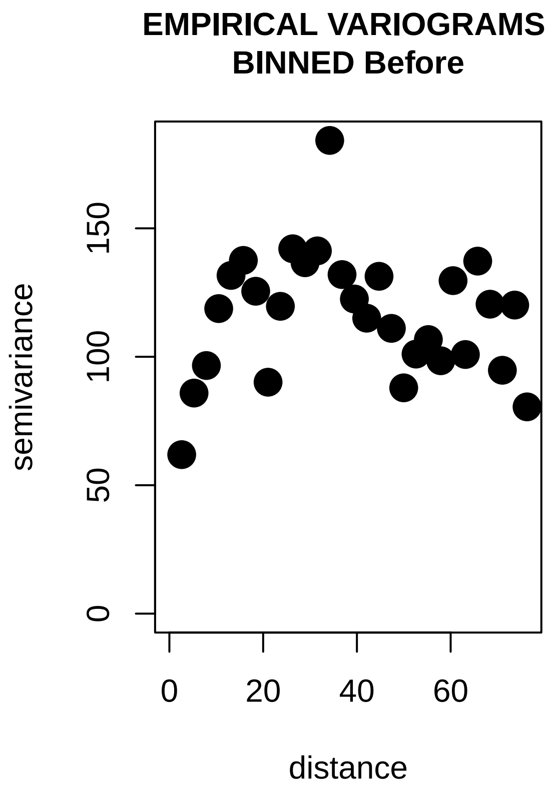
Making the same plots as earlier, to see if the process looks more isotropic.

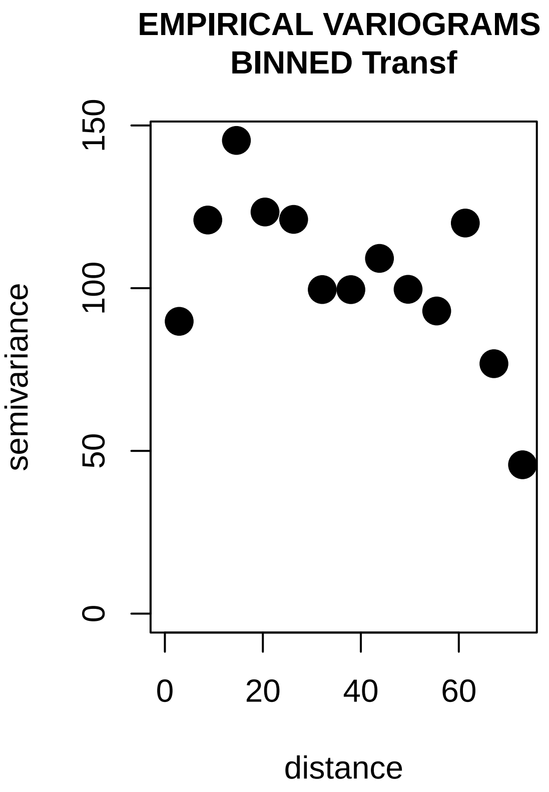
```
maxdist2 <- max(dist(cbind(elevs2.ani[,1],elevs2.ani[,2]))) # Practical rules: lags only up to half of 
cloud2 <- variog(geoelevs2.ani, option = "cloud", estimator.type = "modulus")
bp2 <- variog(geoelevs2.ani, option = "bin", bin.cloud = T,
             pairs.min=30, max.dist=maxdist2/2,
             estimator.type = "modulus") # Not sure about all these arguments.
bin2 <- variog(geoelevs2.ani, option = "bin", pairs.min=30, max.dist=maxdist2/2,
              estimator.type = "modulus")
# Also: pairs.min = 30 since the sample variogram should only be considered for lags that have more tha

par(mfrow=c(1,3))
plot(cloud2,main="CLOUD", cex.main=1, cex.lab=1, cex=2)
plot(bp2, bin.cloud=T, cex.lab=1)
title(main = "BINNED BOXPLOTS", cex.main=1)
plot(bin2, main="EMPIRICAL VARIOGRAM \nBINNED",cex.main=1, cex.lab=1, cex=2, pch=16)
```



Below we compare the sample variograms before and after the transformation.

```
par(mfrow=c(1,2))
plot(bin, main="EMPIRICAL VARIOGRAMS \nBINNED Before",cex.main=1, cex.lab=1, cex=2, pch=16)
plot(bin2, main="EMPIRICAL VARIOGRAMS \nBINNED Transf",cex.main=1, cex.lab=1, cex=2, pch=16)
```





4. Theoretical variograms and estimations of their parameters

In this problem we will propose four theoretical variograms and estimate them via weighted least squares. Later we will select the two variograms that best fit the data and explain the parameters of the chosen variogram. We propose the exponential (with nugget effect), Gaussian (with nugget effect), spherical and Matérn models. Note that this is done with the empirical variaogram 'bin', which is not transformed according to the transformation that was attempted above. In order to find the initial values for optimization of the restricted maximum likelihoods, we used the function below, which is a graphical tool.

```
eyefit(bin, silent = FALSE)
```

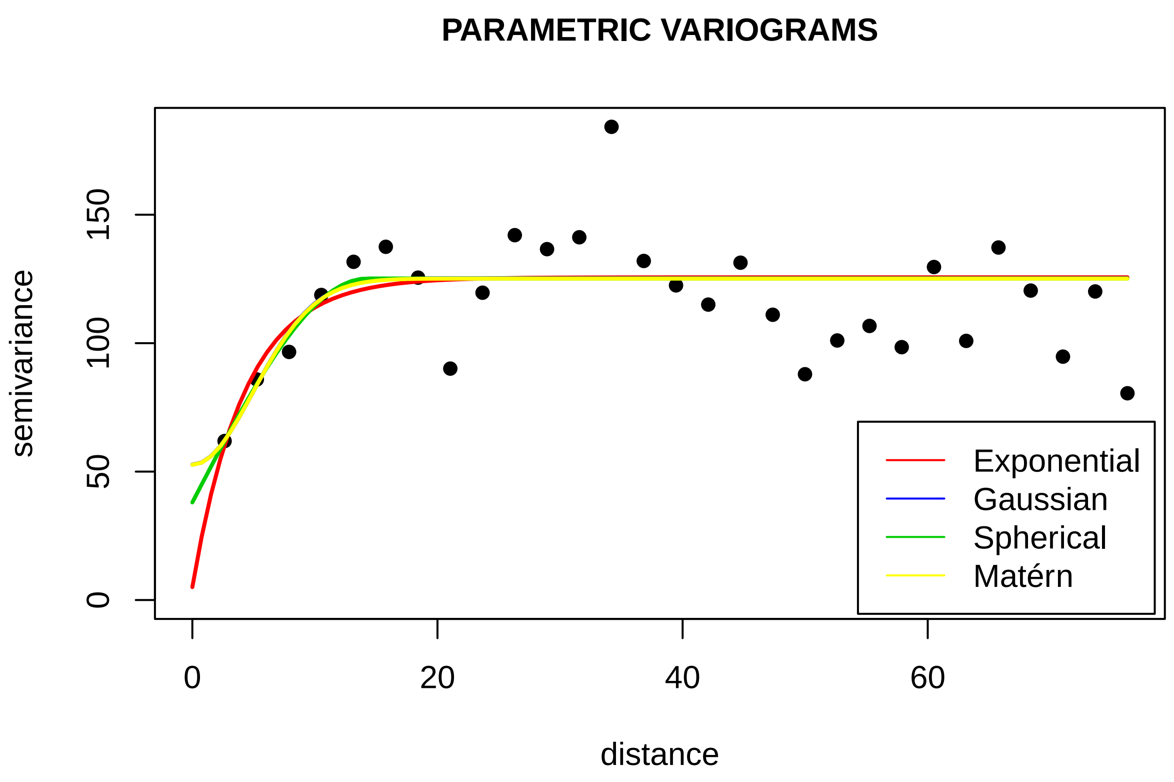
You could check that you arrive at the same values also (or test with other values, perhaps that is better)

The four suggested theoretical variaograms are fitted using weighted least squares, as shown below. The weights used are the ones proposed by Cressie (1985). The initial values for the sill and range are given as the first and second coordinate of the 'ini' vector, respectively. Moreover, we are using models with nugget effect, which means that the nugget is estimated as well, with initial value given by 'nugget' in each function call.

```
# Exponential variogram.
wls1 <- variofit(bin, cov.model = "exponential", ini = c(118,12),
                 fix.nugget = F, nugget =25 , weights="cressie")

# Gaussian variogram.
wls2 <- variofit(bin, cov.model = "gaussian", ini = c(88,8),
                 fix.nugget = F, nugget =42, weights="cressie")

# Spherical.
wls3 <- variofit(bin, cov.model = "spherical", ini = c(130,30),
                  fix.nugget = F, nugget =42, weights="cressie")

# Matérn variogram.
wls4 <- variofit(bin, cov.model = "matern", ini = c(105,16),
                  fix.nugget = F, nugget =42, fix.kappa = FALSE, kappa=0.5,weights="cressie")

# Increase number of bins to estimate the variograms (as done in 'variogramaEstimationScallops3.R')?
# This estimates a new sample variogram I think. Should this be done earlier, in problem 2/3 already pe
```

The parametric variaograms are plotted with the empirical variogram below.

```
par(mfrow=c(1, 1))
plot(bin,  main = "PARAMETRIC VARIOGRAMS",cex.main = 1, pch = 16)
lines(wls1, lwd = 2, col = "red", max.dist = maxdist/2)
lines(wls2, lwd = 2, col = "blue", max.dist = maxdist/2)
lines(wls3, lwd = 2, col = "green3", max.dist = maxdist/2)
lines(wls4, lwd = 2, col = "yellow", max.dist = maxdist/2)
legend(x = "bottomright", inset = 0.01, lty = c(1, 1), col = c("red", "blue", "green3","yellow"),
       legend = c("Exponential", "Gaussian", "Spherical","Matérn"), cex = 1)
```



Selecting the variograms that best fit the data can be done by a combination of inspecting the semivariance plot (qualitative) and comparing the sum of squares from the minimizations. From inspection it looks like the Spherical and the Gaussian/Matérn (they are practically on top of each other) are the two best models. The quantitative measures are summarized in the table below.

```
comparison <- cbind(c("Exponential", "Gaussian", "Spherical","Matérn"),
                    c(summary(wls1)$sum.of.squares[[1]], summary(wls2)$sum.of.squares[[1]],
```

```r
                     summary(wls3)$sum.of.squares[[1]], summary(wls4)$sum.of.squares[[1]]))
colnames(comparison) = c("Model", "Sum of Squares")
knitr::kable(comparison, caption = "Sum of Squares of each of the estimated parametric models.")
```

Table 1: Sum of Squares of each of the estimated parametric models.

Model	Sum of Squares
Exponential	179.417088115306
Gaussian	175.24637336529
Spherical	173.911196359872
Matérn	175.293191829504

From the table we see that the Spherical and the Gaussian models have the smallest sum of squares, which is in accordance with what we thought from the plot above. Hence, we choose the Spherical and Gaussian models. Now, let us explain the parameters from the models.

In general the parameters of the variogram represent the following quantities:

- Nugget: The nugget effect presents micro-scale variation or measurement error. This can be seen graphically as the y-value where the variogram crosses the y-axis.
- Sill: The sill represents the variance of the random field. Note that a quantity called the *partial sill* is defined as $\sigma^2 = \text{sill} - \text{nugget}$.
- Range: The range represents the distance at which the data no longer are autocorrelated. This can be seen graphically by noting the distance, i.e. the x-value, where the variogram stops increasing or becomes approximately parallell to the x-axis.

Now, how about the parameters of the chosen variograms?

The estimated parameters of the two models are

```r
# Spherical.
spherical <- cbind("Partial Sill"=summary(wls3)$spatial.component[[1]],
                   "Range"=summary(wls3)$spatial.component[[2]],
                   "Nugget" = summary(wls3)$nugget.component[[1]])
knitr::kable(spherical, caption = "Estimated Spherical model parameters")
```

Table 2: Estimated Spherical model parameters

Partial Sill	Range	Nugget
87.18164	14.24726	37.99506

```r
# Gaussian.
gaussian <- cbind("Partial Sill"=summary(wls2)$spatial.component[[1]],
                  "phi"=summary(wls2)$spatial.component[[2]],
                  "Nugget" = summary(wls2)$nugget.component[[1]])
knitr::kable(gaussian, caption = "Estimated Gaussian model parameters")
```

Table 3: Estimated Gaussian model parameters

Partial Sill	phi	Nugget
72.54804	7.052085	52.69773

From Table 2 we can see that the predicted sill of the Spherical model is ≈ 125.18 (partial sill + nugget) and the range is ≈ 14.25.

From Table 3 we can see that the predicted sill of the Gaussian model is ≈ 125.25 (partial sill + nugget) and the range is $\sqrt{3}\phi \approx 12.21$, which is the distance to reach 95% of the sill in the Gaussian model.

5. Predictions of elevations along the area of study

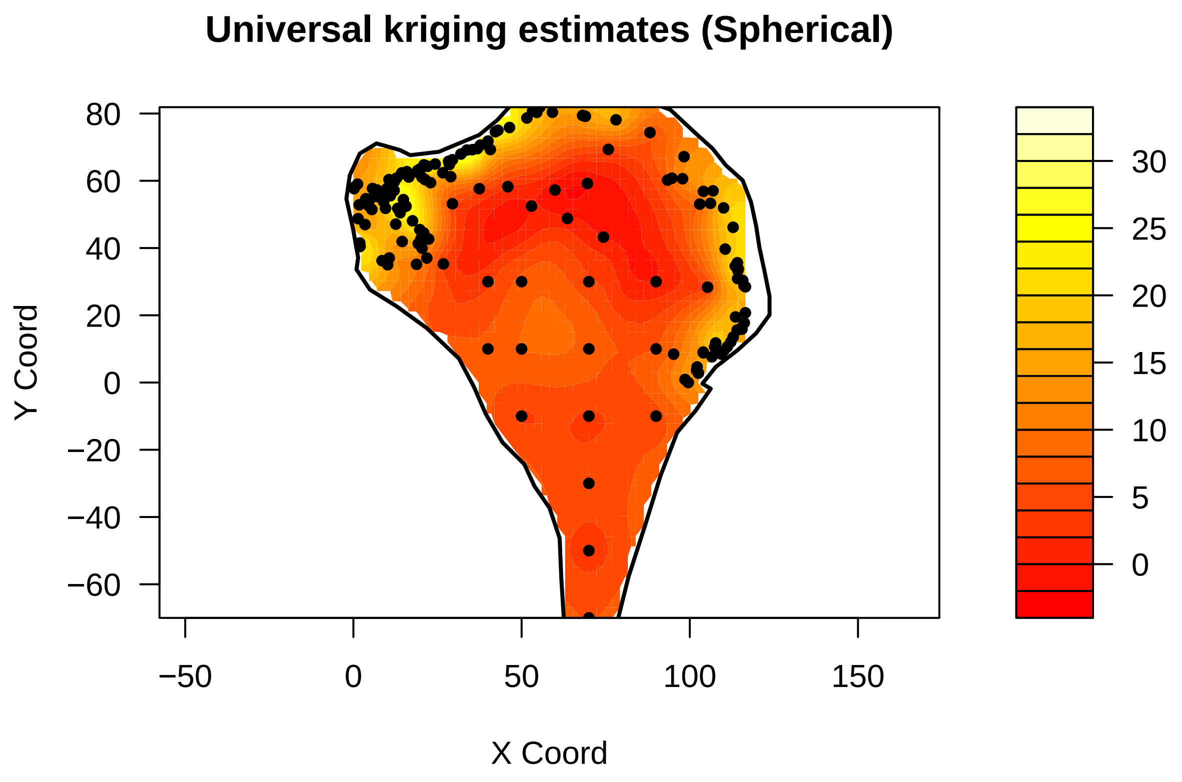
In this final problem we will use kriging to predict elevations along the area of study. This will be done using the two best variograms among the four proposed theoretical variograms in problem 4.

```r
coord.geo <- as.geodata(coordinates)

rnx <- range(geoelevs2$coords[,1])
rny <- range(geoelevs2$coords[,2])
newx.grid <- seq(rnx[1],rnx[2],l=51)
newy.grid <- seq(rny[1],rny[2],l=51)
dsgr.grid <- expand.grid(newx=newx.grid, newy=newy.grid)
```

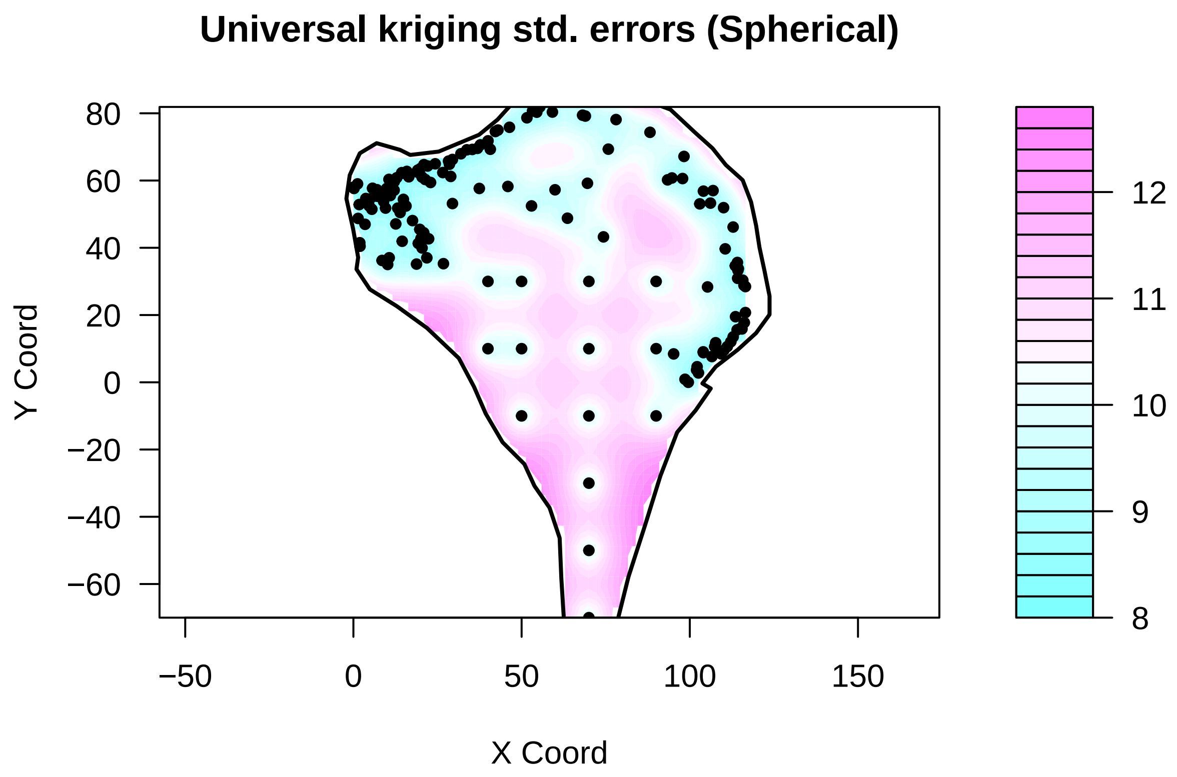
We tried Simple Kriging and Universal Kriging for both the Spherical and the Gaussian variograms. The predictions look very similar, but we think that the Universal Kriging should be the preferred method here, since we have observed a trend in the data earlier. However, the standard errors are the same in both methods, as seen in the plots. Why does this happen?

```r
# Need to estimate the chosen variograms again, on the original data.
bin3 <- variog(geoelevs, option = "bin", pairs.min=30, max.dist=maxdist/2,
                estimator.type = "modulus", uvec=seq(0,maxdist/2,l=30))

# New estimation of Spherical variogram.
wls3.2 <- variofit(bin3, cov.model = "spherical", ini = c(130,30),
                 fix.nugget = F, nugget =42, weights="cressie")

# New estimation of Gaussian variogram.
wls2.2 <- variofit(bin3, cov.model = "gaussian", ini = c(88,8),
                 fix.nugget = F, nugget =42, weights="cressie")

OK.spherical <- krige.conv(geoelevs,coords= geoelevs$coords,
                data=geoelevs$data,locations= dsgr.grid, borders = coordinates[,c(1,2)],
                krige =krige.control(type.krige="OK",
                            obj.m = wls3.2,trend.l= ~coords[,2]+poly(coords[,1],2),
                            trend.d=~coords[,2]+poly(coords[,1],2)))

SK.spherical <- krige.conv(geoelevs,coords= geoelevs$coords,
                data=geoelevs$data,locations= dsgr.grid, borders = coordinates[,c(1,2)],
                krige =krige.control(type.krige="SK",obj.m = wls3.2))

OK.gaussian <- krige.conv(geoelevs,coords= geoelevs$coords,
                data=geoelevs$data,locations= dsgr.grid, borders = coordinates[,c(1,2)],
                krige =krige.control(type.krige="OK",
                            obj.m = wls2.2,trend.l= ~coords[,2]+poly(coords[,1],2),
                            trend.d=~coords[,2]+poly(coords[,1],2)))

SK.gaussian <- krige.conv(geoelevs,coords= geoelevs$coords,
                data=geoelevs$data,locations= dsgr.grid, borders = coordinates[,c(1,2)],
                krige =krige.control(type.krige="SK",obj.m = wls2.2))
```

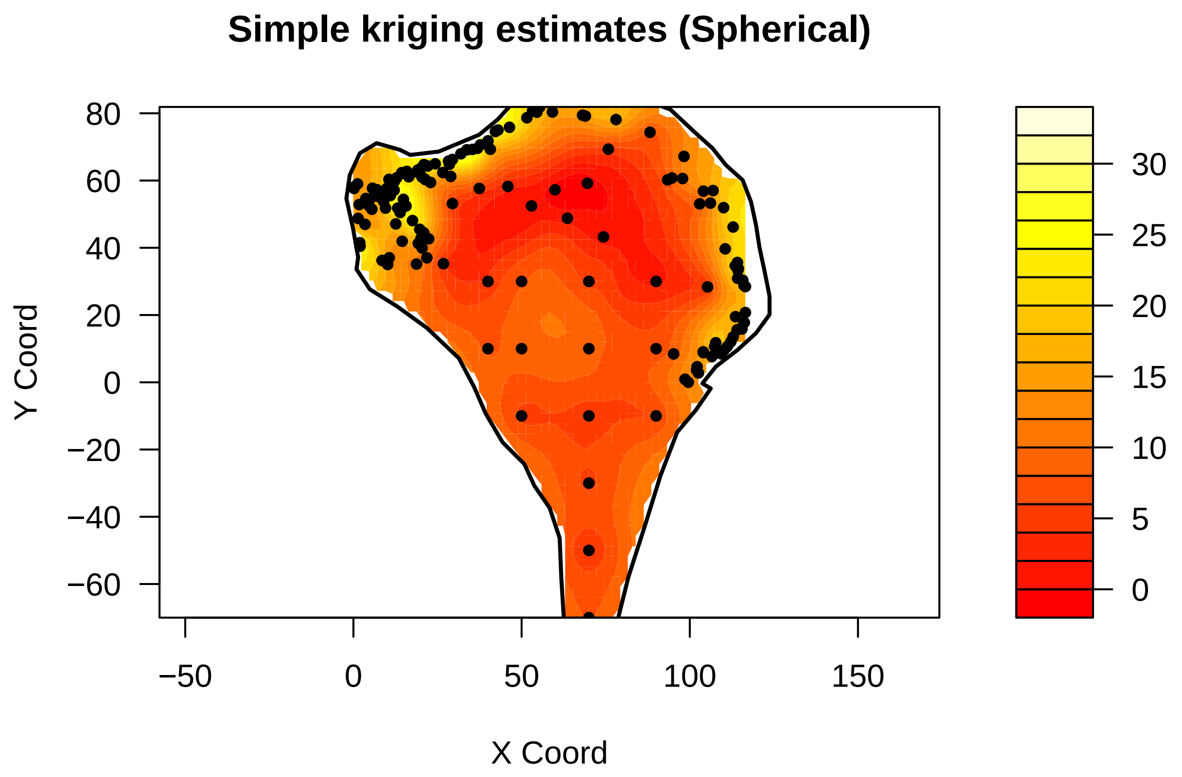
```
# Kriging with Spherical variogram model.
contour(OK.spherical,filled=TRUE,coords.data=geoelevs$coords,color=heat.colors,
        main="Universal kriging estimates (Spherical)")
```



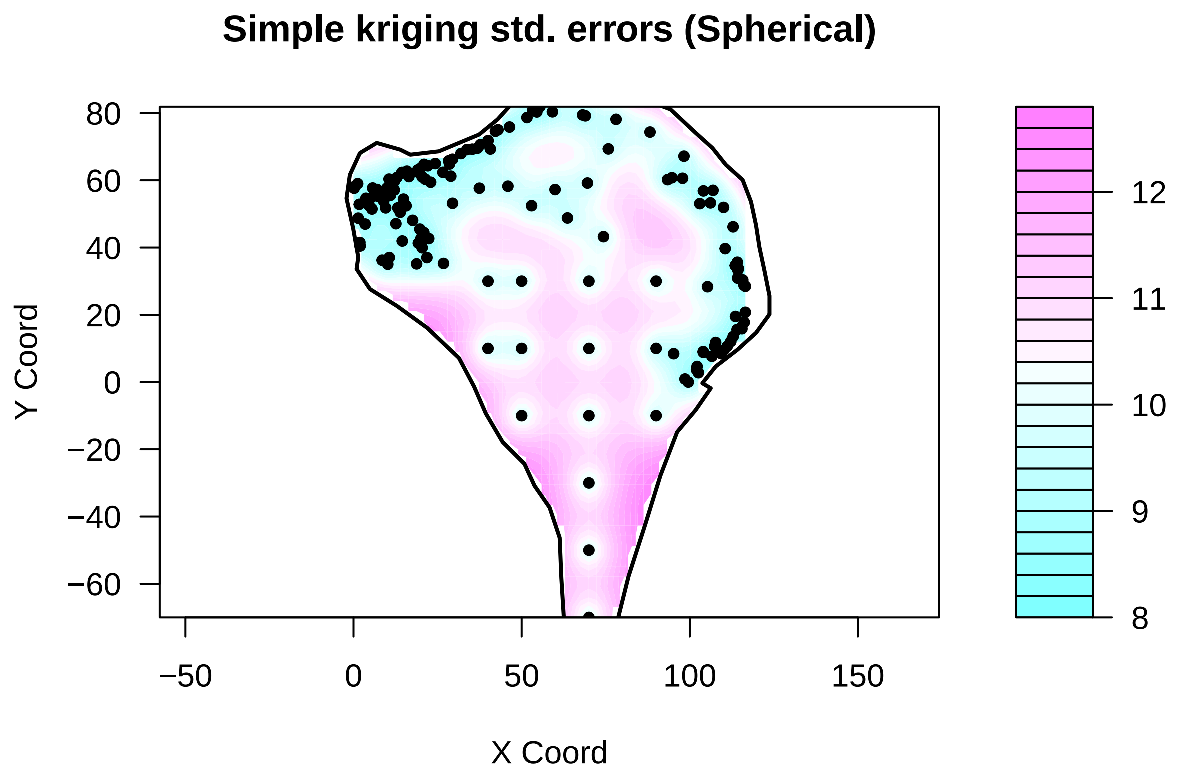
```
contour(OK.spherical,val=sqrt(SK.spherical$krige.var),
        filled=TRUE,coords.data=geoelevs$coords,color=cm.colors,
        main="Universal kriging std. errors (Spherical)")
```



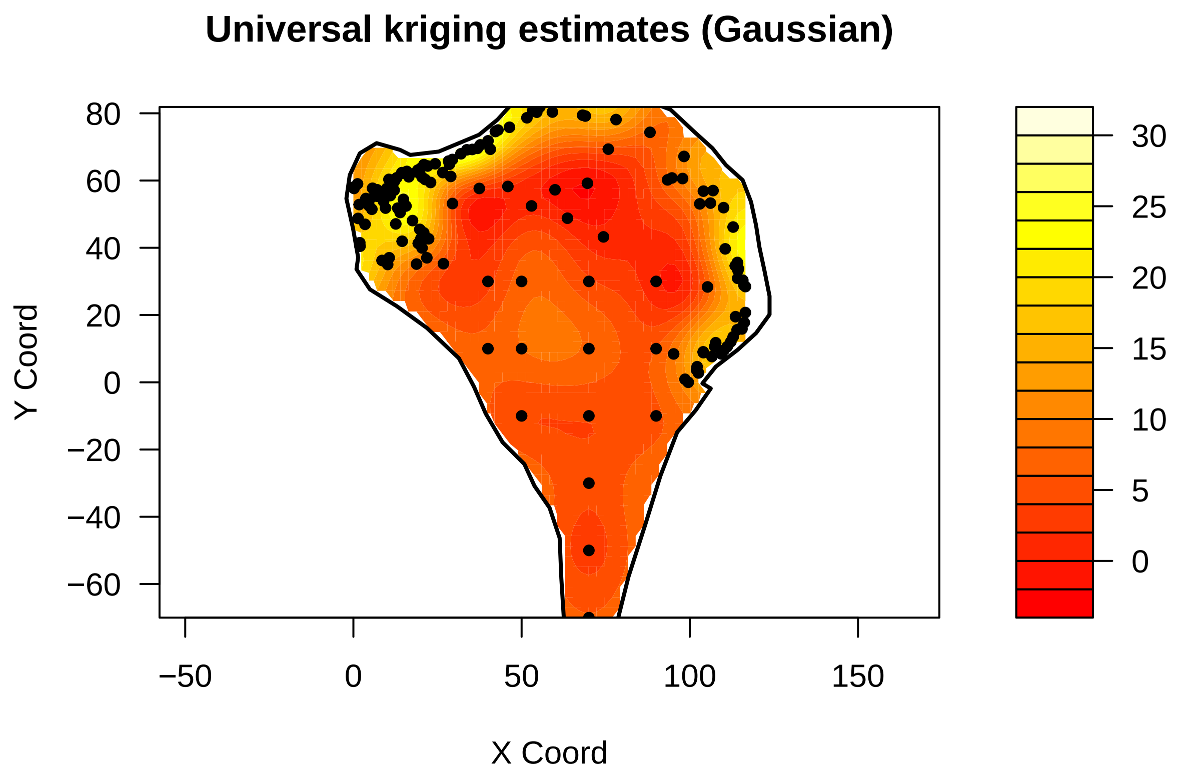
```
contour(SK.spherical,filled=TRUE,coords.data=geoelevs$coords,color=heat.colors,
        main="Simple kriging estimates (Spherical)")
```



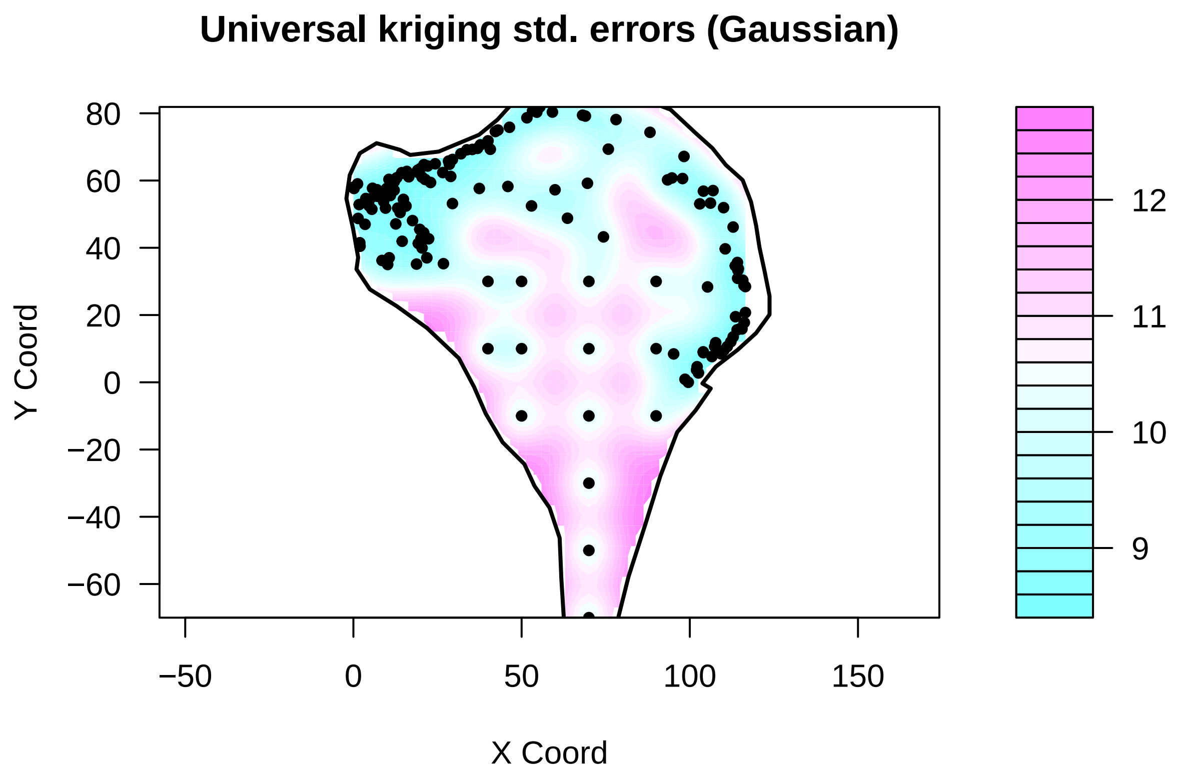
```
contour(SK.spherical,val=sqrt(SK.spherical$krige.var),
        filled=TRUE,coords.data=geoelevs$coords,color=cm.colors,
        main="Simple kriging std. errors (Spherical)")
```



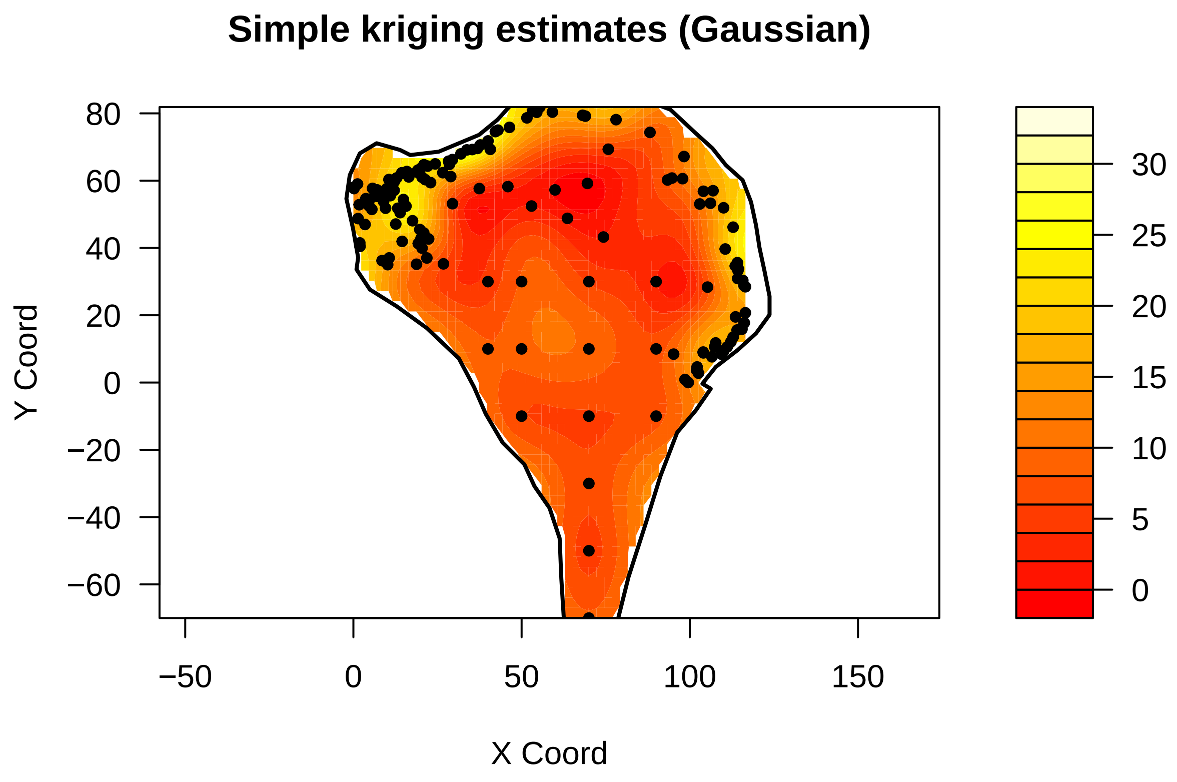
```
# Kriging with Gaussian variogram model.

contour(OK.gaussian,filled=TRUE,coords.data=geoelevs$coords,color=heat.colors,
        main="Universal kriging estimates (Gaussian)")
```



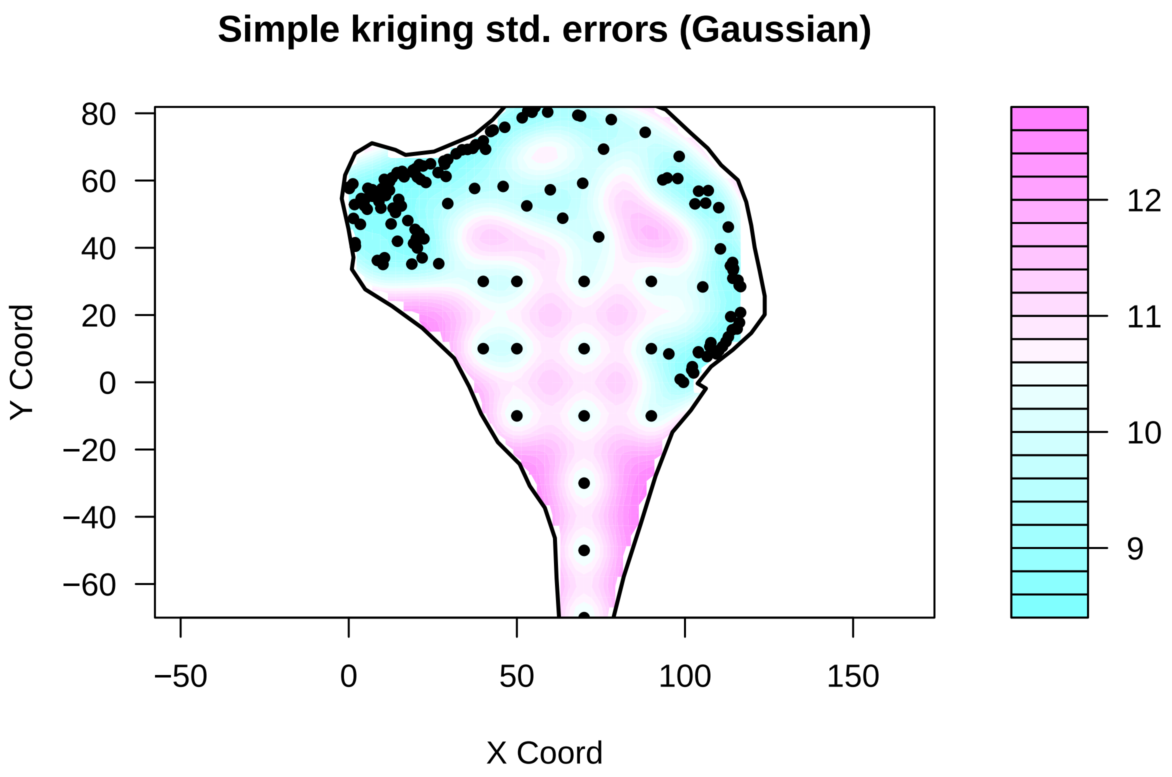
```
contour(OK.gaussian,val=sqrt(SK.gaussian$krige.var),
        filled=TRUE,coords.data=geoelevs$coords,color=cm.colors,
        main="Universal kriging std. errors (Gaussian)")
```



```
contour(SK.gaussian,filled=TRUE,coords.data=geoelevs$coords,color=heat.colors,
        main="Simple kriging estimates (Gaussian)")
```



```
contour(SK.gaussian,val=sqrt(SK.gaussian$krige.var),
        filled=TRUE,coords.data=geoelevs$coords,color=cm.colors,
        main="Simple kriging std. errors (Gaussian)")
```



The best variogram model for kriging will be chosen using cross-validation, as is done below.

```
VC1 <- function(xv){
  mean(xv$error/sqrt(xv$krige.var))
}

VC2 <- function(xv){
  sqrt(mean((xv$error/sqrt(xv$krige.var))^2))
}

VC3 <- function(xv){
  sqrt(mean(xv$error^2))
}

xv.spherical <- xvalid(geoelevs, model = wls3.2, reestimate = F)
xv.gaussian <- xvalid(geoelevs, model = wls2.2, reestimate = F)

cross.validation <- cbind("Value" = c("VC1", "VC2", "VC3"),
                          "Spherical" = c(VC1(xv.spherical), VC2(xv.spherical), VC3(xv.spherical)),
                          "Gaussian"= c(VC1(xv.gaussian), VC2(xv.gaussian), VC3(xv.gaussian)))
```

```
knitr::kable(cross.validation, caption = "Values from cross-validation")
```

Table 4: Values from cross-validation

Value	Spherical	Gaussian
VC1	-0.0165724653146642	-0.015869001868
VC2	0.942066845549418	0.976785569600466
VC3	8.53988603960454	8.90984595202371

The best model has $\text{VC1} \approx 0, \text{VC2} \approx 1$ and low values of VC3. As is apparent from the table above, the models look very similar, but VC1 and VC2 are both better for the Gaussian variogram model. Even though VC3 is better for thte Spherical model, we conclude that the Gaussian model is the best. Another reason to why we conclude this is that the standard errors in the kriging predictions when using the Gaussian model look slightly smaller in the plots given above, when comparing to the Spherical model. Either way, both models are very similar and give very similar predictions.