Lab Exercise #9

The objective of this lab is to try out some basic geostatistical tools with R. Geostatistics is used primarily in the resource and environmental areas for estimation, uncertainty quantification and integrating data with difference volumetric supports, precisions and sources. This lab will introduce you to variograms and kriging for spatial data using the R package **gstat**.

The data for this exercise is the Meuse dataset which is a classical geostatistical data set used frequently by the creator of the **gstat** package 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 data set comprises of four heavy metals (e.g. zinc) measured in the top soil.

```
library(lattice)

library(sp)

data(meuse) #use the system file

coordinates(meuse) <- c("x", "y")

#map the zinc value

data(meuse.grid)

spplot(meuse, "zinc", do.log = T)

#create a raster surface of distance to river

coordinates(meuse.grid) = ~x+y

proj4string(meuse.grid) <- CRS("+init=epsg:28992")

gridded(meuse.grid) = TRUE

spplot(meuse.grid)

image(meuse.grid["dist"])

title("distance to river (red = 0)")
```

Q1. Take a screen shot of the two plots. Do you see any potential correlation between the zinc concentration and distance to river? Explain if any. (10 points)

###Fit a linear regression to model the relationship between zinc and distance to the river. Plot maps with fitted values and with residuals.

```
print(xyplot(log(zinc)~sqrt(dist), as.data.frame(meuse), asp = 1), split = c(1,1,2,1), more = TRUE)
```

```
zn.lm <- lm(log(zinc)~sqrt(dist), meuse)

meuse$fitted.s <- predict(zn.lm, meuse) - mean(predict(zn.lm, meuse))

meuse$residuals <- residuals(zn.lm)

print(spplot(meuse, c("fitted.s", "residuals"), cuts=4, col.regions=grey.colors(5, 0.95, 0.55, 2.2)), split =
c(2,1,2,1))
```

Q2. Take a screen shot of the plots. Describe the pattern of the residuals (10 points)

Usually, interpolation is done on a regular grid. For the Meuse data set, coordinates of points on a regular grid are already defined in the meuse.grid data.frame, and are converted into a SpatialPixelsDataFrame.

```
data(meuse.grid)

coordinates(meuse.grid) <- c("x", "y")

meuse.grid <- as(meuse.grid, "SpatialPixelsDataFrame")


###inverse distance weighted interpolation

install.packages("gstat")

library(gstat)

idw.out <- idw(zinc~1, meuse, meuse.grid, idp = 2.5)

as.data.frame(idw.out)[1:5,]

spplot(idw.out["var1.pred"], main = "zinc inverse distance weighted interpolations")
```

Q3. Take a screen shot of the plot. Modify the idp value to 5 and rerun the above statements. Compare the result against the first one. (15 points)

Variograms are calculated using the function variogram, which takes a formula as its first argument: log(zinc)~1 means that we assume a constant trend for the variable log(zinc).

```
svgm <- variogram(log(zinc) ~ 1, meuse)

svgm

###varigram map

cld <- variogram(log(zinc) ~ 1, meuse, cloud = TRUE)
svgm <- variogram(log(zinc) ~ 1, meuse)
d <- data.frame(gamma = c(cld$gamma, svgm$gamma),
        dist = c(cld$dist, svgm$dist),
        id = c(rep("cloud", nrow(cld)), rep("sample variogram", nrow(svgm)))
        )
xyplot(gamma ~ dist | id, d,
        scales = list(y = list(relation = "free", ylim = list(NULL, c(-.005,0.7)))),
        layout = c(1, 2), as.table = TRUE,
        panel = function(x,y, ...) {
                  if (panel.number() == 2)
                            ltext(x+10, y, svgm$np, adj = c(0,0.5)) #$
                  panel.xyplot(x,y,...)
        },
        xlim = c(0, 1590),
        cex = .5, pch = 3
)
```

Q4. Take a screen shot of the plot. What are the differences between a variogram cloud (top one) and empirical variogram (button one). (20 points)

```
##Cutoff, Lag Width, Direction Dependence
plot(variogram(log(zinc) ~ 1, meuse))
#It simply computes and plots the sample variogram, it does make a number of decisions by default. It decides that direction is ignored: point pairs are merged on the basis of distance, not direction. An alternative is, for example to look in four different angles

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

Q5. Take a screen shot of the plot. Describe and compare the patterns in the four subplots. (10 points)

```
###Variogram Modelling
v <- variogram(log(zinc) ~ 1, meuse)
plot(v)
v <- variogram(log(zinc) ~ 1, meuse)
v.fit <- fit.variogram(v, vgm(1, "Sph", 800, 1)) ##spherical model
plot(v, v.fit, pch = 3)
```

Q6. Take a screen shot of the plot. (10 points)

The following command calculates a directional sample variogram, where directions are binned by direction angle alone. For two point pairs, Z(s) and Z(s+h), the separation vector is h, and it has a direction. Here, we will classify directions into four direction intervals

```
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=3))
```

Q7. Take a screen shot of the plot. (10 points)

```
### Simple, Ordinary, and Universal Kriging
lz.sk <- krige(log(zinc)~1, meuse, meuse.grid, v.fit, beta = 5.9)
lz.ok <- krige(log(zinc)~1, meuse, meuse.grid, v.fit)
lz.uk <- krige(log(zinc)~sqrt(dist), meuse, meuse.grid, v.fit)
plot(lz.sk)
plot(lz.ok)
plot(lz.uk)
```

Q8. Take a screen shot of these plots and describe their differences. (15 points)

```
#Model Diagnostics (cross validation)
sel100 <- sample(1:155, 100)
m.model <- meuse[sel100,]
m.valid <- meuse[-sel100,]
v100.fit <- fit.variogram(variogram(log(zinc)~1, m.model), vgm(1, "Sph", 800, 1))
m.valid.pr <- krige(log(zinc)~1, m.model, m.valid, v100.fit) #[using ordinary kriging]
resid.kr <- log(m.valid$zinc) - m.valid.pr$var1.pred
summary(resid.kr)
resid.mean <- log(m.valid$zinc) - mean(log(m.valid$zinc))
R2 <- 1 - sum(resid.kr^2)/sum(resid.mean^2)
R2
```