# Lab Practice of Seminar on Kriging and Variogram modelling on 2D data

*By Yuanyuan Xia and Amir Mahdi Latifi, Winter 2021, REN R 690*

**Seminar Video:**
https://www.youtube.com/watch?v=GUm48aRQM50&t=1097s&ab_channel=AmirMahdiLatifi

**Background:**
Kriging is a method of spatial interpolation of spatial data, based on prerequisite covariance calculations. In this seminar, Amir has covered Ordinary Kriging applications in geology and mining, as well as the prior variogram calculation and modeling. Yet the applications of Kriging are not limited to geology and mining but with any spatial data (e.g., climate data, geographic distributions of certain species of animal, etc.), Kriging is a powerful tool to estimate the value of the variables of interest with limited available samples. This lab aims at providing some useful instructions on how to implement Kriging with R.

## 1. Load the data, packages and attach the grid

Please download the data set *2D_MV_200Wells.csv* before starting (original could be found at 'https://github.com/GeostatsGuy/GeoDataSets'). A CSV file will be provided at our course webpage.
The data file consists of variables porosity, permeability and acoustic impedance and the X and Y coordinates at the sampling location. Although the data is a 2D set, Kriging can also be applied in 3D data sets. Install the package '*gstat*' (geostatistical methods by Edzer Pebesma), *'sp'* (spatial points addition to regular data frames), and *'plyr'* (splitting, applying, and combining data by Hadley Wickham) for the next steps.

### a) Install packages and library:

```
library(gstat)                             # geostatistical methods by Edzer Pebesma
library(sp)                          # spatial points addition to regular data frames
library(plyr)                                    # manipulating data by Hadley Wickham
```

### b) Specify the grid parameters

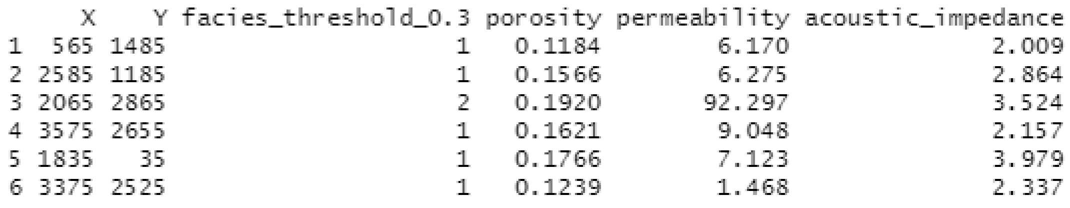
Please download *griddef.txt* for the grid parameters input. You can always change grid parameters based on your own need (the block size that fits your needs best). These parameters define how large your grid cells would be. Smaller grid cells and higher numbers of cells result in an increase in computation time, but also increases the level of details in the final model. For this dataset, the block size is set as 10 meters in X and Y directions (xsize =10, ysize = 10), 400 blocks in each direction (nx = 400, ny = 400), and the origin point of the grid is xmin=5.0 and ymin=5.0 (this is the center location of the first grid cell).

```
nx = 400                              # number of cells in the x direction
ny = 400                              # number of cells in the y direction
xmin = 5.0                            # x coordinate of lower, left cell center
ymin = 5.0                            # y coordinate of lower, left cell center
xsize = 10.0                          # extent of cells in x direction
ysize = 10.0                          # extent of cells in y direction
```

## c) Import the data set

```
mydata = read.csv("2D_MV_200Wells.csv")
head (mydata)
```

The imported data should look like this:



```
     X    Y facies_threshold_0.3 porosity permeability acoustic_impedance
1  565 1485                    1   0.1184        6.170              2.009
2 2585 1185                    1   0.1566        6.275              2.864
3 2065 2865                    2   0.1920       92.297              3.524
4 3575 2655                    1   0.1621        9.048              2.157
5 1835   35                    1   0.1766        7.123              3.979
6 3375 2525                    1   0.1239        1.468              2.337
```

## d) Data Preparation

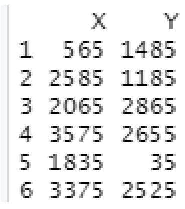
Before starting, our CSV dataframe has to be converted to a series of spatial points for visualization in the next steps. In this example, X and Y coordinates are supposed to be defined here for our data points.

```
class(mydata)                        # confirms that it is a dataframe
coordinates(mydata) = ~X+Y           # indicate the X, Y spatial coordinates
```

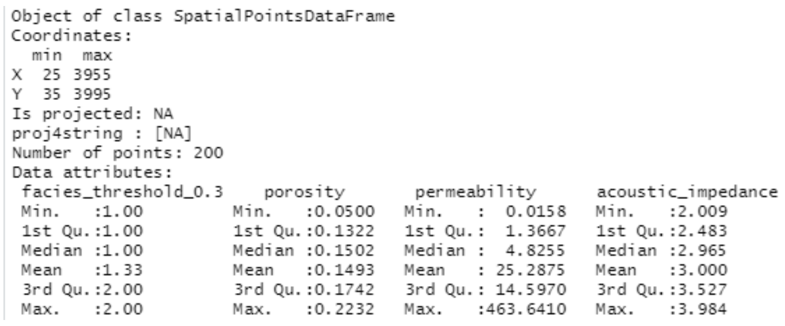
After the conversion, we can further check the summary statistics and do the visualization of the spatial coordinates we created:

```
summary(mydata)                      # confirms a spatial points dataframe
head(coordinates(mydata))            # check the first several coordinates
```

The results for this step should look like this:



```
     X    Y
1  565 1485
2 2585 1185
3 2065 2865
4 3575 2655
5 1835   35
6 3375 2525
```



```
Object of class SpatialPointsDataFrame
Coordinates:
   min  max
X  25 3955
Y  35 3995
Is projected: NA
proj4string : [NA]
Number of points: 200
Data attributes:
 facies_threshold_0.3    porosity        permeability       acoustic_impedance
 Min.   :1.00         Min.   :0.0500   Min.   :  0.0158   Min.   :2.009
 1st Qu.:1.00         1st Qu.:0.1322   1st Qu.:  1.3667   1st Qu.:2.483
 Median :1.00         Median :0.1502   Median :  4.8255   Median :2.965
 Mean   :1.33         Mean   :0.1493   Mean   : 25.2875   Mean   :3.000
 3rd Qu.:2.00         3rd Qu.:0.1742   3rd Qu.: 14.5970   3rd Qu.:3.527
 Max.   :2.00         Max.   :0.2232   Max.   :463.6410   Max.   :3.984
```

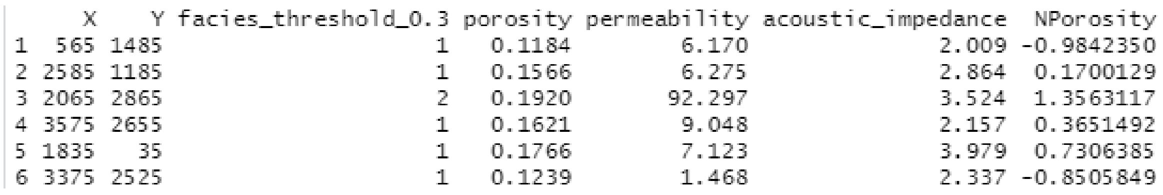
**e) Normal score transformation**

For calculation of the experimental variograms, we often work with Gaussian transformed data to produce more interpretable variograms.

Ashton Shortridge (2008) has written the function to achieve the transformation. Apply the raw data and define it as a vector X, with the function below, R will returns an object with normal score values as a member vector *[my_transform_object]$nscore*.

```
nscore <- function(krig) {
  # Takes a vector of values x and calculates their normal scores. Returns
  # a list with the scores and an ordered table of original values and
  # scores, which is useful as a back-transform table. See backtr().
  nscore <- qqnorm(krig, plot.it = FALSE)$krig  # normal score
  trn.table <- data.frame(krignscore=sort(krig),nscore=sort(nscore))
  return (list(nscore=nscore, trn.table=trn.table))
}
npor.trn = nscore(mydata$porosity)                     # normal scores transform
mydata[["NPorosity"]]<-npor.trn$nscore        # append the normal scores transform
head(mydata)                                                # check the result
```

The NS transformed data should look like this:

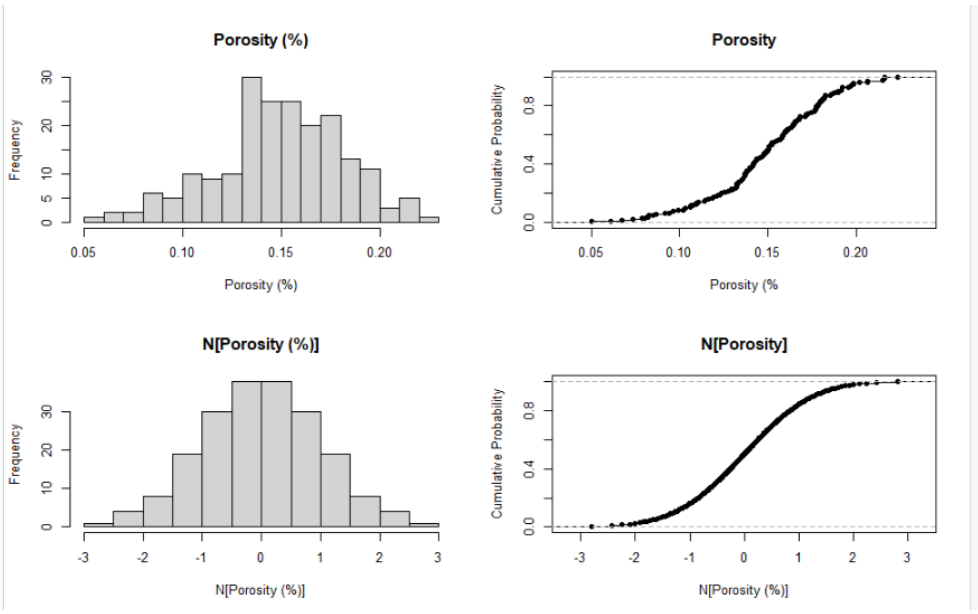


```
     X    Y facies_threshold_0.3 porosity permeability acoustic_impedance  NPorosity
1  565 1485                    1   0.1184        6.170              2.009 -0.9842350
2 2585 1185                    1   0.1566        6.275              2.864  0.1700129
3 2065 2865                    2   0.1920       92.297              3.524  1.3563117
4 3575 2655                    1   0.1621        9.048              2.157  0.3651492
5 1835   35                    1   0.1766        7.123              3.979  0.7306385
6 3375 2525                    1   0.1239        1.468              2.337 -0.8505849
```

Run summary statistics for the normal score transformed data and visualize original and normal scored transformed data to check the outcome of the transformation:

```
summary(mydata$NPorosity)
par(mfrow=c(2,2))
```

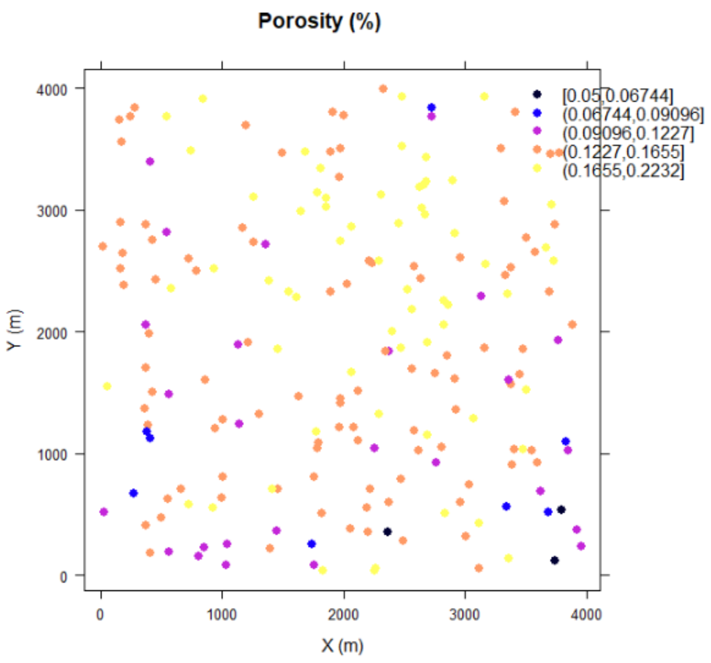
```
hist(mydata$porosity,main="Porosity (%)",xlab="Porosity (%)",nclass = 15)
plot(ecdf(mydata$porosity),main="Porosity",xlab="Porosity (%",ylab="Cumulative
Probability")
hist(mydata$NPorosity,main="N[Porosity (%)]",xlab="N[Porosity (%)]",nclass = 15)
plot(ecdf(mydata$NPorosity),main="N[Porosity]",xlab="N[Porosity(%)]",ylab="Cumulat
ive Probability")
```



## f) Spatial Visualization

Visualize a porosity location map.

```
spplot(mydata,"porosity", do.log = TRUE,
key.space=list(x=.85,y=0.97,corner=c(0,1)),
scales=list(draw=T),xlab = "X (m)", ylab = "Y (m)",main ="Porosity (%)")
```



## 2. Experimental Variogram Calculation and Modelling

First, the experimental variogram must be calculated for the major and minor directions. Usually, to determine these directions we can observe the location map. The spread of highs and lows usually refer to the major direction of continuity. Here we found the major direction is 35°, but it is highly recommended to change this parameter a couple of times to check how the experimental variogram results change. In theory, the points in the major direction should be most continuous. *Cutoff* here refers to the maximum acceptable distance to pair points. *Width* refers to the tolerance for distances between pairs of points.

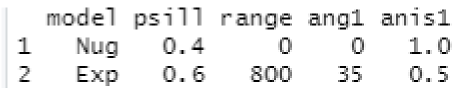
```
por.vg.035 = variogram (NPorosity~1, mydata, cutoff=3000, width=500, alpha=35.0,
tol.hor=22.5) # 035 directional
por.vg.125 = variogram(NPorosity~1,mydata,cutoff = 3000,width =500,alpha =
125.0,tol.hor=22.5) # 125 directional
```

Variogram modeling (two structures) is implemented with the following command:

```
    por.vm.ani <- vgm(psill = 0.6, "Exp", 800, anis = c(035, 0.5),nugget=0.4)
    por.vm.ani                         # check the variogram model parameters
```

This is the two structures variogram model, *psill* refers to the contribution of the first structure, 800 is the range of the first structure and nugget effect refers to the starting points on y axis of the variogram model.

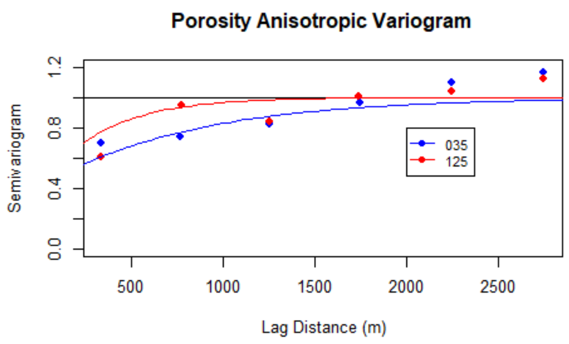
The results is displayed below:



```
  model psill range ang1 anis1
1   Nug   0.4     0    0   1.0
2   Exp   0.6   800   35   0.5
```

Visualize the variogram model:

```
    name = c("035","125")                         # make name matrix
    color = c("blue","red")                       # make color matrix
    plot(por.vg.035$dist,por.vg.035$gamma,main="Porosity Anisotropic Variogram",xlab="
    Lag Distance (m) ",ylab=" Semivariogram ",pch=16,col=color[1],ylim=c(0,1.2))
    points(por.vg.125$dist,por.vg.125$gamma,pch=16,col=color[2]) abline(h = 1.0)
    unit_vector = c(sin(35*pi/180),cos(35*pi/180),0) # unit vector for 035 azimuth
    vm.ani.035 <-
    variogramLine(por.vm.ani,maxdist=3000,min=0.0001,n=100,dir=unit_vector,covariance=
    FALSE) # model at 035
    lines(vm.ani.035$dist,vm.ani.035$gamma,col=color[1]) # include variogram model
    unit_vector = c(sin(55*pi/180),-1*cos(35*pi/180),0) # unit vector for 125 azimuth
    vm.ani.125 <-
    variogramLine(por.vm.ani,maxdist=3000,min=0.0001,n=100,dir=unit_vector,covariance=
    FALSE) # model at 125
    lines(vm.ani.125$dist,vm.ani.125$gamma,col=color[2]) # include variogram model
    legend(2000,.8,name, cex=0.8, col=color,pch=c(16,16,16),lty=c(1,1,1)) # add legend
```



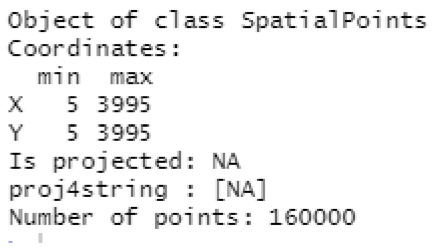
In the above figure, points represent the experimental variogram calculation results, while the lines (blue and red for the major and minor directions) represent the model for the calculated results.

## 3. Ordinary Kriging Estimation and Visualization

### a) Adding coordinates to grid cells:

The *addcoord* command below is included to attach coordinates to the grid cells which we defined earlier.

```
addcoord <- function(nx,xmin,xsize,ny,ymin,ysize) {
   coords = matrix(nrow = nx*ny,ncol=2)
   ixy = 1
   for(iy in 1:nx) {
       for(ix in 1:ny) {
        coords[ixy,1] = xmin + (ix-1)*xsize
        coords[ixy,2] = ymin + (iy-1)*ysize
        ixy = ixy + 1
  }
} # Function is written by Michael Pyrcz, 2008
coords.df = data.frame(coords)
colnames(coords.df) <- c("X","Y")
coordinates(coords.df) =~X+Y
return (coords.df)
}
```



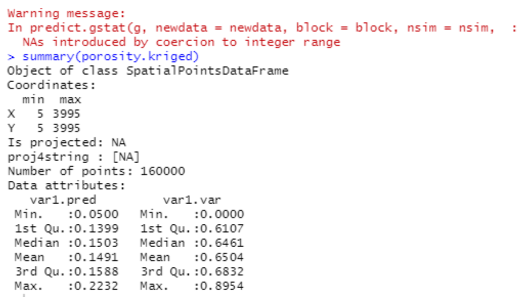
```
Object of class SpatialPoints
Coordinates:
  min  max
X   5 3995
Y   5 3995
Is projected: NA
proj4string : [NA]
Number of points: 160000
```

From the results, we can see a block model is specified based on our data range with a 4000*4000 grid in XY plane.

b) Ordinary Kriging without searching limitations

```
porosity.kriged = krige(porosity~1, krig, coords, model = por.vm.ani,maxdist = Inf,nmin = 0,omax=Inf) # ordinary Kriging
summary(porosity.kriged)
```

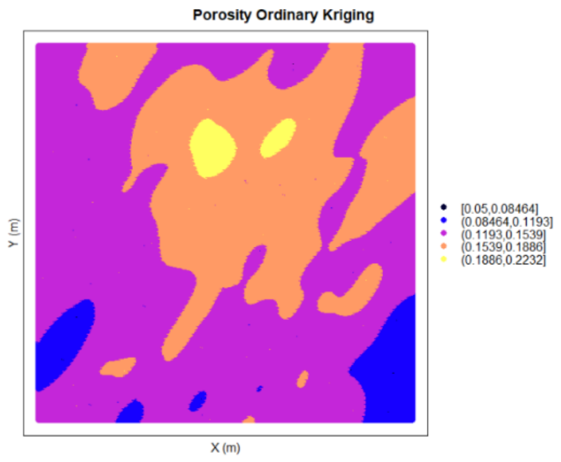
Result is shown below:



```
Warning message:
In predict.gstat(g, newdata = newdata, block = block, nsim = nsim,  :
  NAs introduced by coercion to integer range
> summary(porosity.kriged)
Object of class SpatialPointsDataFrame
Coordinates:
  min  max
X   5 3995
Y   5 3995
Is projected: NA
proj4string : [NA]
Number of points: 160000
Data attributes:
   var1.pred         var1.var
 Min.   :0.0500   Min.   :0.0000
 1st Qu.:0.1399   1st Qu.:0.6107
 Median :0.1503   Median :0.6461
 Mean   :0.1491   Mean   :0.6504
 3rd Qu.:0.1588   3rd Qu.:0.6832
 Max.   :0.2232   Max.   :0.8954
```

**Visualize the Kriging results:**

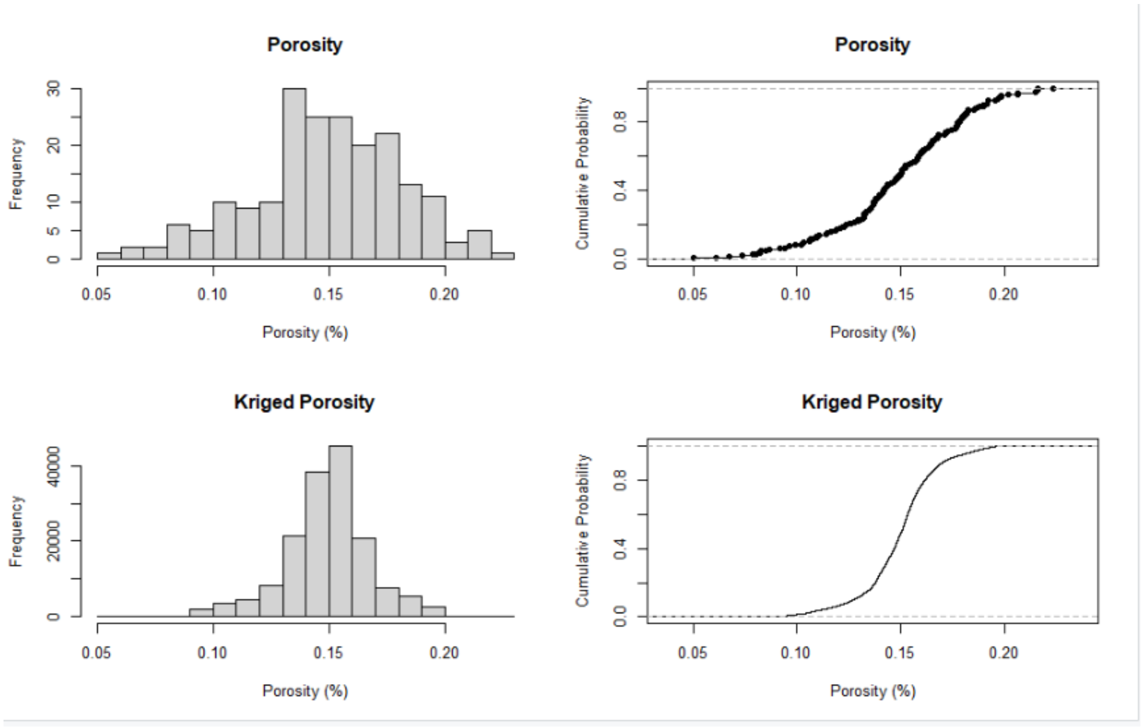
```
spplot(porosity.kriged["var1.pred"],main = "Porosity Ordinary Kriging", key.space = "right", xlab = "X (m)", ylab = "Y (m)")
```



When we compare the original porosity data and ordinary Kriging porosity data (in terms of their histograms), it is obvious that the Kriging estimation smooths the data (lower standard deviation).

Compare the histogram and CPP:

```
par(mfrow=c(2,2))
hist(mydata$porosity,main="Porosity",xlab="Porosity (%)",nclass = 15)
plot(ecdf(mydata$porosity),main="Porosity",xlab="Porosity (%)",ylab="Cumulative
Probability")
hist(porosity.kriged$var1.pred,main="Kriged Porosity",xlab="Porosity (%)",nclass =
15) plot(ecdf(porosity.kriged$var1.pred),main="Kriged Porosity",xlab="Porosity
(%)",ylab="Cumulative Probability")
```



### c) Ordinary Kriging with limited search

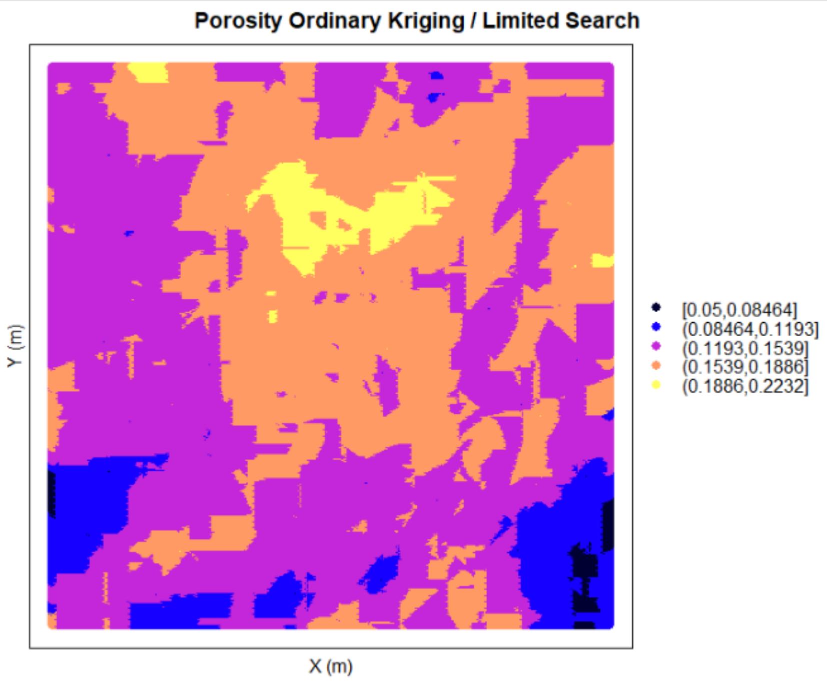
The searching radius will impact the results, as a large searching radius will increase the number of samples used in the estimation of each grid cell. If a search radius limit is implemented in Ordinary Kriging, the results will have higher estimation variances (lower accuracy) and would most likely produce sudden changes in values in areas with limited sampling.

Here we set up maximum searching radius as 800 meters, minimum number of pairs used is 3 and maximum pairs used per octant is 1.

```
maxdist = 800                                  # maximum distance to look for
data
nmin = 3                                      # minimum number of data for an
estimate
omax = 1                                    # maximum number of data per octant
```

After setting up the searching parameters, we use the following commands to interpolate and visualize the results:

```
porosity.kriged.sp = krige(porosity~1, mydata, coords, model = por.vm.ani,maxdist = maxdist,nmin = nmin,omax=omax) # ordianry Kriging
spplot(porosity.kriged.sp["var1.pred"],main = "Porosity Ordinary Kriging / Limited Search", key.space = "right", xlab = "X (m)", ylab = "Y (m)")
```



Comparing to the results of unrestricted ordinary Kriging with reasonable searching limits produces less smooth results. With the same command in the previous section, we can produce the histogram and CPP for limited searching radius Kriging as well. It is highly encouraged to try it out and compare the data smoothness.

Please feel free to contact Yuanyuan (yuanyua1@ualberta.ca) or Amir (amirmahdi.latifi@ualberta.ca) if you have any further questions or feedbacks. Thank you for the participation in the seminar and lab.

Referenced resources:
https://github.com/GeostatsGuy