Lab 4: Geostatistical Data Analysis in R

Objectives

- Simulate a Gaussian spatial process
- Estimate empirical semi-variograms
- Fit models to semi-variograms
- Conduct spatial prediction by Kriging

Part I: Kriging with Simulated Data

In part I, we will simulate a Gaussian spatial process and extract a subset as our working data. We will treat the subset of data as the sample data in practice, which are usually obtained during field work. We then estimate its semivariogram and conduct Kriging.

1. The GeoR package

The `geoR` package is a powerful R package containing many useful geostatistics functions. It is capable of much more than the simple cases we will consider here. For more information, see the additional online documentation about the package (as usual). Note that there is an entire reference manual available online:

`http://cran.r-project.org/web/packages/geoR/geoR.pdf`

Now, install and load the related package into your R workspace (along with the two others).

```
> install.packages("geoR",repos = "http://cran.r-project.org")
> install.packages("maptools",repos = "http://cran.r-project.org")
> install.packages("maps",repos = "http://cran.r-project.org")
> install.packages("mvtnorm",repos = "http://cran.r-project.org")
```

```
> library(geoR)
> library(maps)
> library(mvtnorm)  # package for simulation
> rm(list=ls())     # clear workspace
```

2. Simulating a Gaussian spatial process

Simulation is a powerful tool for analyzing spatial processes. Let's simulate a Gaussian spatial process on a regular square domain encompassing Ohio. First, obtain the limits of our spatial domain:

```
> oh.range=map("state","ohio",plot=FALSE)$range
> rx=oh.range[2]-oh.range[1]     # range in x direction
> ry=oh.range[4]-oh.range[3]  # range in y direction
```

Next, define the spatial points (locations) at which to simulate the Gaussian process. In order to find a set of regular locations, you should use the `expand.grid` function. Since the range in x direction (rx) and y direction (ry) are almost the same, we expand the grid using equal number of sequences in both direction. Here, we create a set of regular locations of 30×30.

```
> xg=seq(oh.range[1],oh.range[2], length.out=30)
> yg=seq(oh.range[3],oh.range[4], length.out=30)
> ohgrid.locs=expand.grid(xg,yg)
> map("state","ohio")
> points(ohgrid.locs,pch=20)
> ntot=dim(ohgrid.locs)[1]
```

In this case, let the Gaussian spatial process be defined as:

$$\boldsymbol{Y} \sim N(\boldsymbol{\mu},\Sigma(\boldsymbol{\theta}))$$

where the covariance matrix is defined by the exponential covariance model (without a nugget effect):

$$[\Sigma(\boldsymbol{\theta})]_{i,j} = \theta_1 \exp\{-d_{ij}/\theta_2\},$$

where the $[\ ]_{i,j}$ notation refers to the (i, j)-th element of the matrix. The expression above contains sill and range parameters (θ_1 and θ_2, respectively) and also d_{ij}, the distance between each (i,j) pair of locations. Thus, in order to define the covariance matrix for the spatial process, we need to specify the model parameters and find the distances between all locations. Note that our model for this process assumes stationarity and isotropy.

We can use the function called `dist()` in R to get a matrix of pairwise distances between all points in our data set.

```
> distmat=as.matrix(dist(ohgrid.locs))
> max_dist=max(distmat)
```

Now create the covariance matrix in terms of the exponential model.

```
> theta1=2     # we are making these values up in
> theta2=1     # order to specify a true covariance matrix.
> Sig=theta1*exp(-distmat/theta2)
```

To visualize the spatial structure you have specified in this covariance matrix, plot the 'true' covariogram at 20 equally spaced spatial lags:

```
> plot( seq(0,max_dist,length.out=20),
        theta1*exp(-seq(0,max_dist,length.out=20)/theta2),
        type="b",ylab="cov",xlab="distance" )
```

Now simulate a spatial random process from the Gaussian model given a mean of 5 and the covariance matrix constructed just above:

```
> y=rmvnorm(1,matrix(5,ntot,1),Sig)   # takes some time
> image(matrix(y,length(xg),length(yg)),x=xg,y=yg,
        col= rev(rainbow(100,start=0,end=.7)))
> map("state","ohio",lwd=3,add=TRUE)
> box()   # add a boundary box to the plot
```

Note that the color of each grid cell in the image represents the value of the process at the point located in the grid-cell center. (These are not data defined for the whole grid cell; we are only plotting them this way for visual convenience.)

3. Use a subset of the simulated process as working data

A process is rarely observed with such complete spatial coverage. In practice, we usually only obtain a sample of the data at visited locations during fieldwork. In order to analyze this process, suppose only a subset of the data is available. Specifically, construct a dataset comprised of only 40% of the originally simulated process:

```
> idxkeep=sort(sample(1:ntot,round(0.4*ntot)))
> ymask=matrix(0,ntot,1)
> ymask[idxkeep,]=1
> image(matrix(y,length(xg),length(yg)),x=xg,y=yg,
         col=rev(rainbow(100,start=0,end=.7)))
> image(matrix(ymask,length(xg),length(yg)),x=xg,y=yg,
          col=c("white","transparent"),add=TRUE)
> map("state","ohio",add=TRUE,lwd=3)
> box()
```

4. Estimate the empirical semi-variogram

First, create a `geodata` object using the subset of simulated data (a `geodata` object is a list with two components: `coords` and `data`).

```
> n=length(idxkeep)
> ydat=matrix(y[idxkeep],n,1)
> dat.locs=ohgrid.locs[idxkeep,]
> ygeodat=as.geodata(cbind(dat.locs,ydat))
```

Then estimate the empirical semi-variogram of the sample data using function `variog`. The `max.dist` is a parameter in `variog` that defines the maximum distance for the variogram estimation. Pairs of locations separated for distance larger than this value are ignored for the variogram calculation. As a rule of thumb, the largest lag distance is set to be no greater than half of the maximum distance between sample points.

```
> distsample=as.matrix(dist(dat.locs))      # distance between sample pairs
> maxd = max(distsample)/2
> y.v=variog(ygeodat,max.dist=maxd)         # estimate empirical semi-variogram
> y.v$beta.ols    # estimated trend of the sample data
```

The true semivariogram can be calculated as

```
> vt = theta1-theta1*exp(-y.v$u/theta2)
```

We can plot the empirical semi-variogram and true semi-variogram together. To get a better view, you need to set the `xlim` and `ylim` of plot area.

```
> xmax = max(y.v$u,maxd)     # xmax to set xlim for plot
> ymax = max(y.v$v,vt)       # ymax to set ylim for plot

> plot(y.v, xlim = c(0,xmax), ylim = c(0,ymax) )      # empricial semivariogram
> lines(y.v$u, vt,type="b",pch=20)                    # add true semivariogram
```

Ask yourself how does your empirical semi-variogram compare with the truth? Is your estimate (i.e., `y.v$beta.ols`) of the trend reasonable (recall that the true trend was equal to 5)? Do the empirical semi-variogram estimates match the truth? Usually, there exists difference between the truth and that estimated from sample data.

In addition, the function `variog4` provides the empirical semi-variogram in each of 4 principle directions. Note also that you can calculate a particular directional semi-variogram by setting the direction and tolerance arguments in the `variog` function.

```
> plot(variog4(ygeodat,max.dist=maxd))
```

Does the plot above suggest that the true process is isotropic?

5. Fit an exponential semi-variogram model

In this exercise, we will use the empirical estimates from the reduced dataset and weighted least squares to fit a parametric model. In order to fit the model, we need to specify initial values for sill and range. If the initial values are not specified, `variofit` will use the default search. It usually works well. In the following, we won't specify the initial values. If you cannot get a reasonable fit, please try to specify initial values. Please use `help(variofit)` to see more details.

```
> y.v.wls=variofit(y.v, cov.model="exponential", wei="cressie")
> y.v.wls
> plot(y.v ,xlim=c(0,xmax), ylim=c(0,ymax))     # empirical semivariogram
> lines(y.v.wls,col=2)                           # fitted semivariogram
> lines(y.v$u,vt,type="b",pch=20)                # true semivariogram
```

Ask yourself how do the estimated spatial parameters compare with the true values (recall that the true sill was 2 and the true range parameter was 1)? Note that `sigmasq` and `phi` in the output of `y.v.wls` correspond to `theta1` and `theta2` in this Lab.

6. Ordinary kriging

Kriging is an exact predictor. Therefore, there is no need to predict the sampled data. We use `ohgrid.locs[-idxkeep,]` to get the prediction locations. In the following, we will do prediction using ordinary kriging:

```
> pred.locs=ohgrid.locs[-idxkeep,]   # predict un-sampled locations only will save time
> y.krig.ok=krige.conv(ygeodat,loc=pred.locs,

krige=krige.control(type.krige="ok",obj.m=y.v.wls))
> ypred.ok=matrix(0,ntot,1)
> ypred.ok[idxkeep,]=ydat
> ypred.ok[-idxkeep,]=y.krig.ok$predict
> ypredvar.ok=matrix(0,ntot,1)
> ypredvar.ok[-idxkeep,]=y.krig.ok$krige.var
```

In the commands above, the predicted values at the "unobserved" locations are collated with the 'observed' values and stored in the vector `ypred.ok`. Likewise, the prediction variances are stored into a vector, `ypredvar.ok`, corresponding to the full set of locations (observed and unobserved). These vectors can be transformed to matrices and then represented graphically as images:

```
> par(mar=c(4,3,2,0.5)) # set plot margin
> op = par(mfrow=c(2,2), pty="s")

> image(matrix(y,length(xg),length(yg)),
           x=xg,y=yg,col=rev(rainbow(100,start=0,end=.7)),
         main="Full Dataset")
> map("state","ohio",lwd=2,add=TRUE)
> box()
> image(matrix(y,length(xg),length(yg)),x=xg,y=yg,
           col=rev(rainbow(100,start=0,end=.7)),main="Sample Data")
> image(matrix(ymask,length(xg),length(yg)),
           x=xg,y=yg,col=c("white","transparent"),add=TRUE)
> map("state","ohio",lwd=2,add=TRUE)
> box()

> image(matrix(ypred.ok,length(xg),length(yg)),
       x=xg,y=yg, col=rev(rainbow(100,start=0,end=.7)),
       main="Prediction(OK)")
> map("state","ohio",lwd=2,add=TRUE)
> box()

> image(matrix(sqrt(ypredvar.ok),length(xg),length(yg)),
        x=xg,y=yg,col=rev(rainbow(100,start=0, end=.7)),
        main="Prediction Standard Errors (OK)")
> image(matrix(1-ymask,length(xg),length(yg)),
        x=xg,y=yg,col=c("white","transparent"), add=TRUE)
> map("state","ohio",lwd=2,add=TRUE)
> box()

> par(mfrow = c(1,1))
```

The above codes generate a map of the simulated data (full dataset), the sample data, the prediction, and the prediction standard errors. Compare the predicted map with the simulated data to get a visual impression of the accuracy of the prediction.

Assignment I

- Generate a Gaussian spatial random process on a regular grid covering Iowa State with a known mean that you will specify. Use a true exponential covariance model to define the true covariance structure. Make it interesting in the sense that there should be a reasonable amount of spatial dependence.
- Provide the covariogram map and report the model and parameters you chose.
- Provide an image plot of the simulated spatial process with map overlay.
- Use only 30% of the original realization as continuous spatial data (sample it randomly as done in procedure 3). Plot the data (the reduced subset) as an image with a map overlay.
- Fit an appropriate model to the empirical semi-variogram. Plot the empirical, fitted, and true semi-variograms together. How does your empirical semi-variogram compare with the truth? How do the estimated spatial parameters compare with the true values? Is the assumption of isotropy valid here? Should it be?
- Conduct ordinary Kriging and plot four maps in one figure: simulated map, sampled map, prediction map, predicted standard error. Provide the figure in your map.

Part II: Kriging with Real Data

We will study the heavy metal concentration in a flood plain of the river Meuse, near the village of Stein, the Netherlands. Field data were collected by Ruud van Rijn and Mathieu Rikken; compiled for R by Edzer Pebesma; description extended by David Rossiter. Please type `?meuse` in R to get detailed information.

7. Load data

We will use two libraries: `geoR` and `maptools`. The dataset named `meuse` is provided with `sp` package, when you load `geoR`, the dependent library `sp` will also be loaded. Cadmium ppm data will be used for Kriging.

```
> library(geoR)
> library(maptools)

> rm(list=ls())
> data(meuse)                              # data provided with sp package
> ?meuse                                   # information about the dataset

> coords = cbind(meuse$x,meuse$y)                   # read coordinates
> coordinates(meuse) = coords
> bubble(meuse, "cadmium")            # plot points with cadmium column
```

Loading the dataset `meuse.grid` for locations at which Kriging prediction will be made. In order to plot the kriging results on an irregular grid, we need to convert the grid into `SpatialPixelsDataFrame`.

```
> data(meuse.grid)
> Cd.df = SpatialPixelsDataFrame(points = meuse.grid[c("x", "y")],
                                         data = meuse.grid)
> coords.grid = cbind(meuse.grid$x,meuse.grid$y)     # read coordinates
> coordinates(meuse.grid) = coords.grid


> plot(meuse.grid)     # grid locations to predict
```

8. Estimate the empirical semi-variogram

The `geoR` package has its own spatial data object. To get the data into the required format, you can read it directly from a file using the `read.geodata` command (see `?geoR`) or you can use the `as.geodata` command if you already have the data in a matrix or data frame. In the latter case, you can use:

```
> Cdgeodat=as.geodata(cbind(meuse$x,meuse$y,meuse$cadmium))
> distmat=as.matrix( dist(cbind(meuse$x,meuse$y)) )
> maxd = max(distmat)/2
> maxd

> y.v=variog(Cdgeodat,max.dist= maxd)
> y.v$beta.ols
```

Plot empirical semi-variogram:

```
> xmax = max(y.v$u,maxd)
> ymax = max(y.v$v)
> plot(y.v,xlim = c(0,xmax), ylim = c(0,ymax) )
```

Fit an Exponential Semi-Variogram Model

```
> y.v.wls=variofit(y.v, ini.cov.pars=c(10,1000), cov.model="exponential",
wei="cressie")
> y.v.wls
> plot(y.v,xlim = c(0,xmax), ylim = c(0,ymax))
> lines(y.v.wls,col=2)
```

9. Kriging

(1) Ordinary Kriging

```
> y.krig.ok=krige.conv(Cdgeodat,loc=coords.grid,
                       krige=krige.control(type.krige="ok",obj.m=y.v.wls))
```

```
> ypred.ok = y.krig.ok$pred
> yVar.ok = y.krig.ok$krige.var
> Cd.df$OK_pred = ypred.ok
> Cd.df$OK_se = sqrt(yVar.ok)
```

(2) Plot prediction

First, set the plot color scheme:

```
> maxCd = ceiling(max(ypred.ok))   # maxCd = 13
> minCd = floor(min(ypred.ok))     # minCd = 0
> bluepal = colorRampPalette(c("azure3", "blue"))
> brks = seq(minCd,maxCd,2)
> cols = bluepal(length(brks) - 1)

> maxSe = ceiling(max(Cd.df$OK_se))   # maxSe = 4
> minSe = floor(min(Cd.df$OK_se))     # minSe = 2
> brks.se = seq(minSe,maxSe,0.3)
> cols.se = bluepal(length(brks.se) - 1)
```

Then plot the Kriging prediction:

```
> dev.new(width=8, height=4) # create new plot space with specific size
> par(mfrow = c(1, 2), mar=c(1,1,2,1), pty = "s")
> image(Cd.df, "OK_pred", col = cols)
> symbols(coords,circles=meuse$cadmium*5,fg="black",inches=F,add=T)
> legend("topleft", fill=cols, legend=leglabs(brks), bty="n", cex=0.8)
> title(main = "Cd - Ordinary Kriging")
> box()
```

Plot the Kriging standard errors:

```
> image(Cd.df, "OK_se", col = cols.se)
> symbols(coords,circles=meuse$cadmium*0+20,fg="black",inches=F,add=T)
> legend("topleft", fill=cols.se, legend=leglabs(brks.se),bty="n",cex=0.8)
> title(main = "Cd - Kriging Error(OK)")
> box()
> setwd("C:/WorkSpace/Geog8102/Lab4") # suppose you've created the folder
> dev.copy(jpeg,"krigfig.jpg",width=8,height=4,units="in",res=300)
> dev.off()
```

Assignment II

- Repeat Part II with another heavy metal in data `meuse`. Use `names(meuse)` to find out what types of heavy metals there are in the dataset. Use `help(meuse)` to see more information about the dataset.
- Provide a bubble map of the selected heavy metal.
- Plot the empirical and fitted semi-variograms together. Provide the map in your report.
- Provide the prediction map (with ordinary Kriging) and Kriging error map.