2019

STA C273 Final Project

UNEMPLOYMENT IN BARCELONA, SPAIN

CYNTHIA LAI

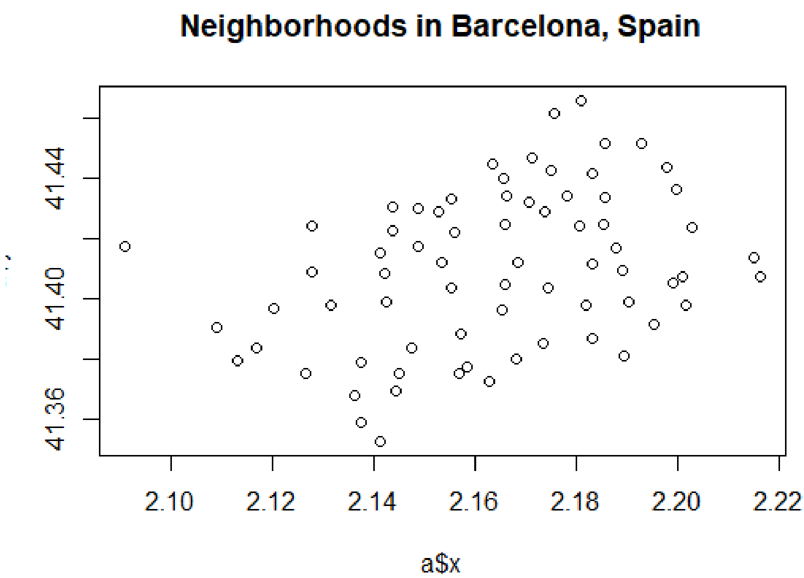
Cynthia Lai

Description of Data

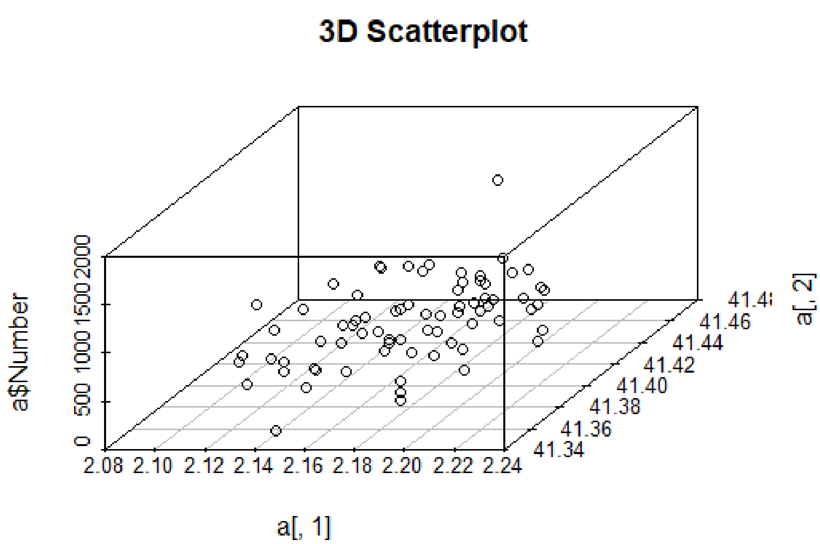
This dataset is one of many Barcelona data sets from Kaggle. I chose the unemployment data set to work with. The data set includes 99 different neighborhoods within Barcelona and the associated count of people registered unemployed.

I subset the data to avoid repeating locations so more specifically, it includes female demand in non-overlapping neighborhoods in January 2017.

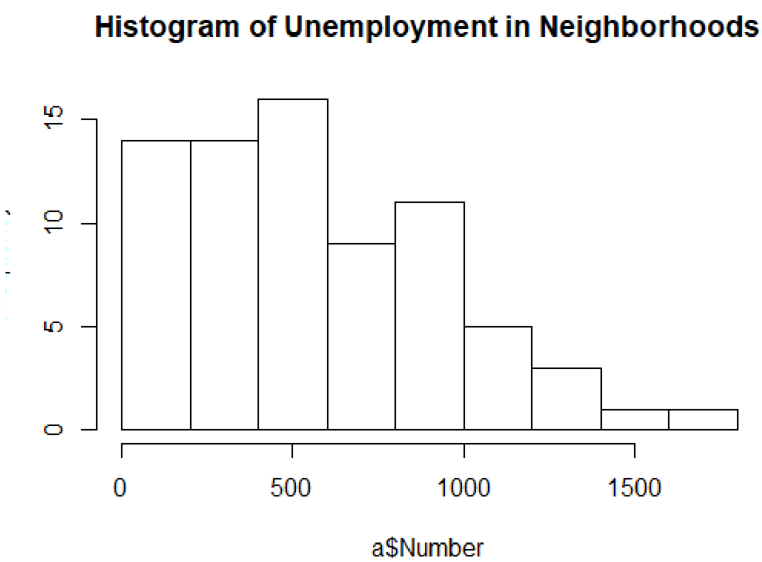
Data Exploration



This is a plot of the original longitude and latitude coordinates for different neighborhoods in Barcelona Spain. They seem to be relatively spaced out.



From this plot, there does not appear to be much noticeable trend in the data.

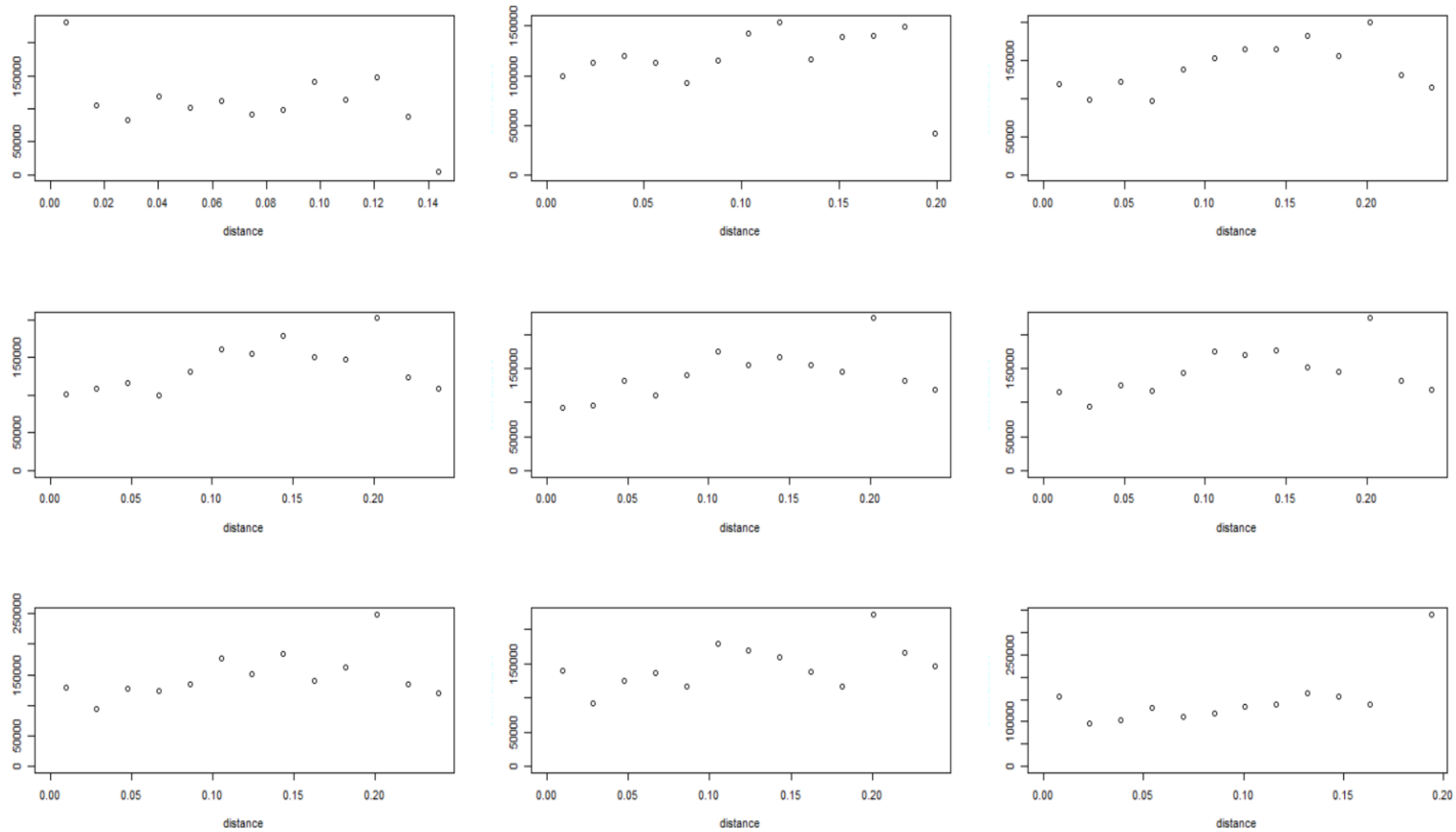


This is the distribution of unemployment numbers across Barcelona. It appears to be skewed right with a couple high values on the right end of the tail.

Geometric Anisotropy

I wanted to test whether the data set was isotropic.

Cynthia Lai

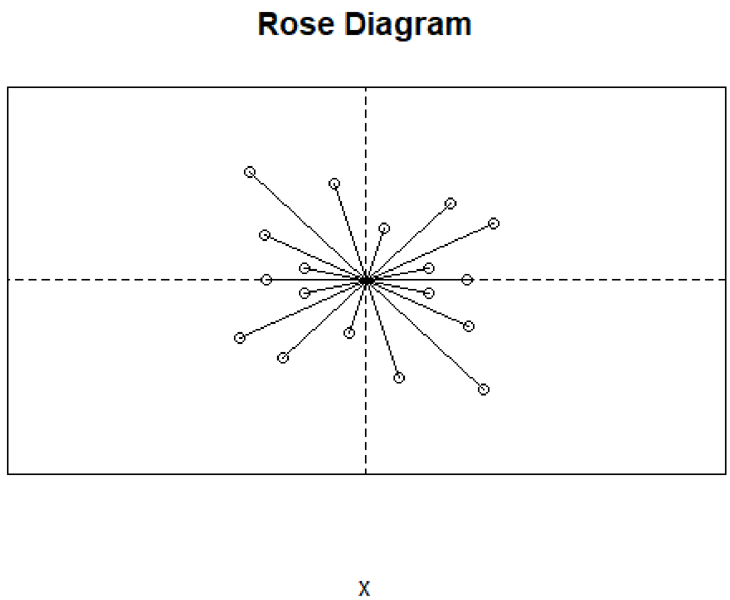


Above are the 9 variograms at different angular tolerances

(0, pi/9, pi/4.5, pi/3, pi/2.25, pi/18, pi/6, pi/3.6, pi/2.571)

We can see that the ranges are different across the angles so we proceed to transform the data to become isotropic.

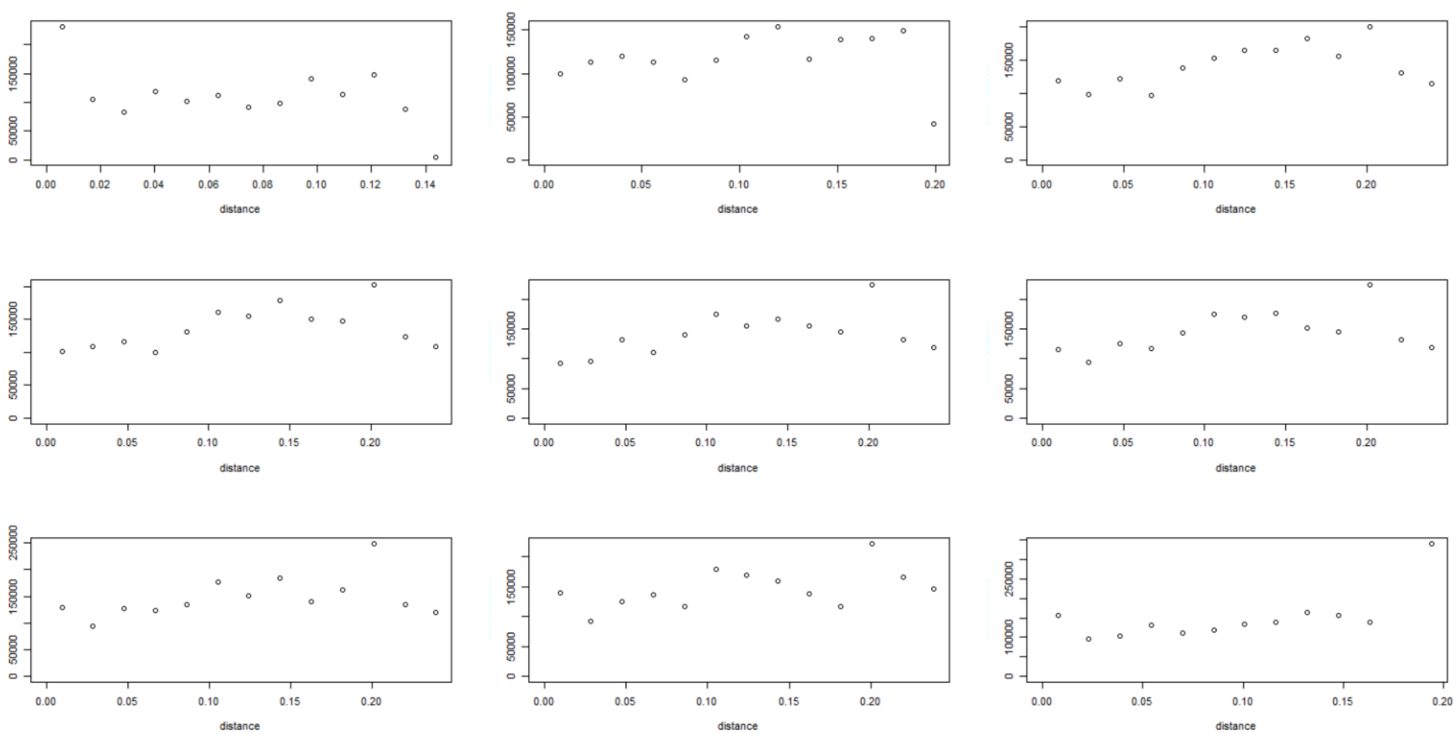
Cynthia Lai



This is the rose diagram associated with the angles and the ranges. It is elliptical.

Rtheta	Rl
[,1] [,2] [1,] 0.8660254 0.5000000 [2,] -0.5000000 0.8660254	[,1] [,2] [1,] 1 0.0 [2,] 0 3.5

After transforming the data, these are the new variograms. We can see that the ranges are approximately the same now.



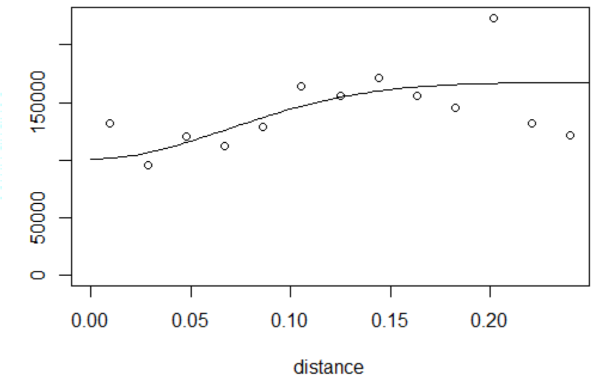
Cynthia Lai

Cross Validation

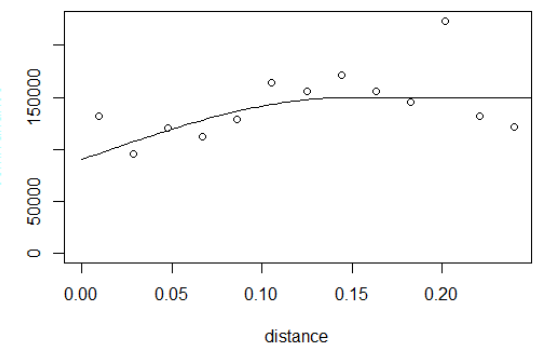
I tried 3 different fits.

a) Gaussian, Cressie

b) Gaussian, equal



c) Spherical, equal



PRESS values

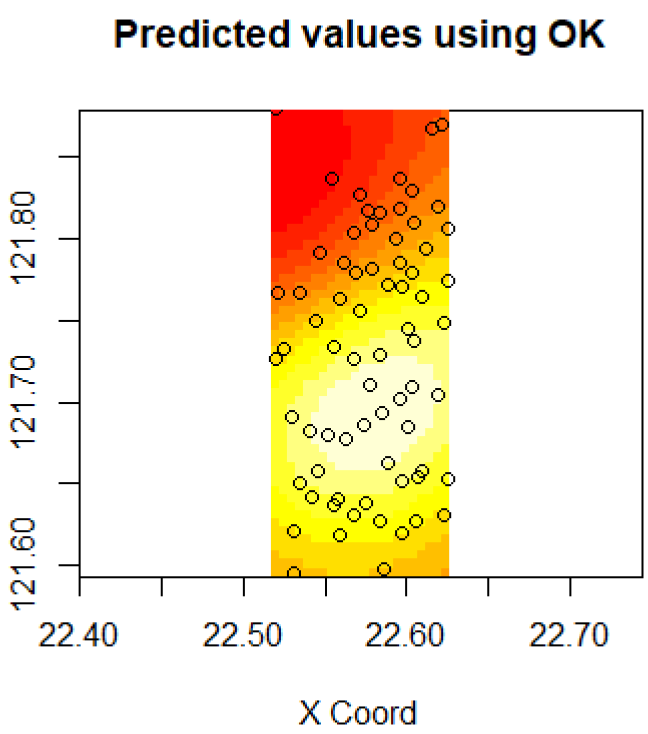
A	B	C
122651.7	116158.7	121398.6

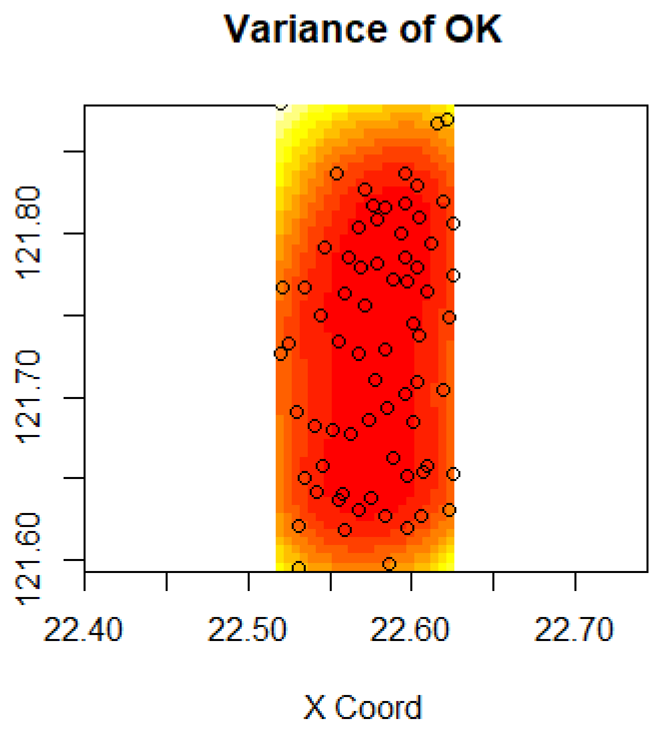
From these numbers, the best model is B. We proceed with using that fit.

Cynthia Lai

Kriging

I used ordinary kriging over a grid. Because there appeared to be no trend earlier in the 3D scatterplot, I did not try universal kriging.





There seems to be high values predicted around the center-bottom and lower values surrounding that area.

Variances appear to be low throughout.

Cynthia Lai

Code Appendix

```
# DATA PREPROCESSING
# data from Kaggle
# https://www.kaggle.com/xvivancos/barcelona-data-sets
dat = read.csv("unemployment.csv")

Subs1 = subset(dat, (dat$Month == "January"))
Subs1 = subset(Subs1, (Subs1$Year == "2017"))
Subs1 = subset(Subs1, Subs1$Gender == "Female")
Subs1 = subset(Subs1, Subs1$Demand_occupation == "Registered unemployed")

a = Subs1[,c(5,6,9)]

library(ggmap)
addresses = paste0(a$Neighborhood.Name, ", Barecelona, Spain")
##################################################################
#key =
##################################################################
# https://stackoverflow.com/questions/52565472/get-map-not-passing-the-api-key-http-
status-was-403-forbidden/52617929#52617929
register_google(key = key)

# https://www.shanelynn.ie/massive-geocoding-with-r-and-google-maps/
getGeoDetails <- function(address){
  #use the gecode function to query google servers
  geo_reply = geocode(address, output='all', messaging=TRUE)
  #now extract the bits that we need from the returned list
  answer <- data.frame(lat=NA, long=NA, accuracy=NA, formatted_address=NA,
address_type=NA)

  #return Na's if we didn't get a match:
  if (geo_reply[2]$status != "OK"){
    return(answer)
  }
  #else, extract what we need from the Google server reply into a dataframe:
  answer$lat <- geo_reply$results[[1]]$geometry$location$lat
  answer$long <- geo_reply$results[[1]]$geometry$location$lng
  if (length(geo_reply$results[[1]]$types) > 0){
    answer$accuracy <- geo_reply$results[[1]]$types[[1]]
  }
  answer$address_type <- paste(geo_reply$results[[1]]$types, collapse=',')
  answer$formatted_address <- geo_reply$results[[1]]$formatted_address
  return(answer)
}
#initialise a dataframe to hold the results
geocoded <- data.frame()
# find out where to start in the address list (if the script was interrupted before):
startindex <- 1

# Start the geocoding process - address by address. geocode() function takes care of
query speed limit.
for (ii in seq(startindex, length(addresses))){
  print(paste("Working on index", ii, "of", length(addresses)))
  #query the google geocoder - this will pause here if we are over the limit.
```

```
  result = getGeoDetails(addresses[ii])
  result$index <- ii
  #append the answer to the results file.
  geocoded <- rbind(geocoded, result)
}
#now we add the latitude and longitude to the main data
data$lat <- geocoded$lat
data$long <- geocoded$long

a$x = geocoded$long
a$y = geocoded$lat

b = a[,c(4,5,3,1,2)]
write.csv(b, file = "dat.csv")


###################################################################
###################################################################
setwd("./w19/sta273/project")

a = read.csv("dat.csv")
a = a[c(2:6)]

## remove points too close together -- messes up kriging
n = nrow(a)
x <- as.matrix(cbind(a$x, a$y))
x1 <- rep(rep(0,n),n)
dist <- matrix(x1,nrow=n,ncol=n)  #the distance matrix

for (i in 1:n){
  for (j in 1:n){
    dist[i,j]=((x[i,1]-x[j,1])^2+(x[i,2]-x[j,2])^2)^.5
  }
}

selected = c()
vals = c()
for (i in 1:n){
  for (j in 1:n){
    if (dist[i,j] < 0.0005) {
      if (i != j) {
        selected = c(selected, i, j)
        vals = c(vals, dist[i,j])
      }
    }
  }
}
remove = c(34, 35, 74)
a = a[-remove,]


plot(a$x, a$y, main="Neighborhoods in Barcelona, Spain")

library(scatterplot3d)
scatterplot3d(a[,1], a[,2], a$Number, main="3D Scatterplot")
```

Cynthia Lai

```
hist(a$Number, main="Histogram of Unemployment in Neighborhoods")


library(gstat)
library(geoR)
library(sp)

b <- as.geodata(a)

### GEOMETRIC ANISOTROPY
#Compute the variogram for the following directions:
var1 <- variog(b, dir=pi/2, tol=pi/4, max.dist=100, estimator.type = "modulus")
var2 <- variog(b, dir=pi/2.57, tol=pi/4, max.dist=100, estimator.type = "modulus")
var3 <- variog(b, dir=pi/3.6, tol=pi/4, max.dist=100, estimator.type = "modulus")
var4 <- variog(b, dir=pi/6, tol=pi/4, max.dist=100, estimator.type = "modulus")
var5 <- variog(b, dir=pi/18, tol=pi/4, max.dist=100, estimator.type = "modulus")
var6 <- variog(b, dir=0.944*pi, tol=pi/4, max.dist=100, estimator.type = "modulus")
var7 <- variog(b, dir=0.833*pi, tol=pi/4, max.dist=100, estimator.type = "modulus")
var8 <- variog(b, dir=0.722*pi, tol=pi/4, max.dist=100, estimator.type = "modulus")
var9 <- variog(b, dir=0.611*pi, tol=pi/4, max.dist=100, estimator.type = "modulus")


#Plot the variograms:
par(mfrow=c(3,3))
plot(var1);plot(var2);plot(var3)
plot(var4);plot(var5);plot(var6)
plot(var7);plot(var8);plot(var9)

par(mfrow=c(1,1))

theta <- c(0, pi/9, pi/4.5, pi/3, pi/2.25, pi/18, pi/6, pi/3.6, pi/2.571)
ranges <- c(0.3, 0.2, 0.5, 0.5, 0.3, 0.55, 0.7, 0.4, 0.2)

x1 <- cos(theta[1:5])*ranges[1:5]
y1 <- sin(theta[1:5])*ranges[1:5]

x2 <- ranges[6:9]*sin(theta[6:9])
y2 <- -ranges[6:9]*cos(theta[6:9])

x11 <- -x1
y11 <- -y1

x22 <- -x2
y22 <- -y2

plot(x1,y1, xlim=c(-1,1), ylim=c(-1,1), xaxt="n", yaxt="n",
     ylab="y", xlab="x", main="Rose Diagram")
points(x11,y11)
points(x2,y2)
points(x22,y22)

segments(x1,y1, x11, y11)
segments(x2,y2, x22, y22)
```

```
segments(0, -34.8, 0, 34.8, lty=2)
segments(-28, 0, 28, 0, lty=2)



t1 <- cos(theta[7])
t2 <- sin(theta[7])
xxx <- c(t1,t2,-t2,t1)

#Create the theta matrix:
Rtheta <- matrix(xxx,nrow=2,ncol=2,byrow=TRUE)

#Ratio of the two major axes:
l <- max(ranges)/min(ranges)
yyy <- c(1,0,0,l)

#Create the l matrix:
Rl <- matrix(yyy, nrow=2, ncol=2, byrow=TRUE)

#Old coordinates:
xy <- as.matrix(cbind(a$x, a$y))

#New coordinates:
xynew <- Rl %*% Rtheta %*% t(xy)

a2 = as.data.frame(t(xynew))
a2$Number = a$Number
names(a2) = c("x","y","Number")
b2 = as.geodata(a2)

#Compute the variogram for the following directions:
var1 <- variog(b2, dir=pi/2, tol=pi/4, max.dist=.25, estimator.type = "modulus")
var2 <- variog(b2, dir=pi/2.57, tol=pi/4, max.dist=.25, estimator.type = "modulus")
var3 <- variog(b2, dir=pi/3.6, tol=pi/4, max.dist=.25, estimator.type = "modulus")
var4 <- variog(b2, dir=pi/6, tol=pi/4, max.dist=.25, estimator.type = "modulus")
var5 <- variog(b2, dir=pi/18, tol=pi/4, max.dist=.25, estimator.type = "modulus")
var6 <- variog(b2, dir=0.944*pi, tol=pi/4, max.dist=.25, estimator.type = "modulus")
var7 <- variog(b2, dir=0.833*pi, tol=pi/4, max.dist=.25, estimator.type = "modulus")
var8 <- variog(b2, dir=0.722*pi, tol=pi/4, max.dist=.25, estimator.type = "modulus")
var9 <- variog(b2, dir=0.611*pi, tol=pi/4, max.dist=.25, estimator.type = "modulus")


#Plot the variograms:
par(mfrow=c(3,3))
plot(var1);plot(var2);plot(var3)
plot(var4);plot(var5);plot(var6)
plot(var7);plot(var8);plot(var9)

par(mfrow=c(1,1))


b = b2; a = a2
```

Cynthia Lai

```
#Compute variogram:
var1 <- variog(b, estimator.type = "modulus", max.dist = 0.25); plot(var1)

c0 = 90000
c1 = 60000
alpha = 0.1

fit1 <- variofit(var1, cov.model="gau", ini.cov.pars=c(c1,alpha),
                 fix.nugget=FALSE, nugget=c0, weights="cressie")
plot(var1);lines(fit1, lty=1)

c0 = fit1$nugget
fit1$cov.pars
c1 = fit1$cov.pars[1]
alpha = fit1$cov.pars[2]


###############################################
# CROSS VALIDATION
# Gaussian Cressie
x_val1 <- xvalid(b, model=fit1)

sum(x_val1$error^2)/nrow(a)

c0 = 90000
c1 = 60000
alpha = 0.1
# Gaussian Equal weights
fit2 <- variofit(var1, cov.model="gau", ini.cov.pars=c(c1,alpha),
                 fix.nugget=FALSE, nugget=c0)
plot(var1); lines(fit2,lty=1)
x_val1 <- xvalid(b, model=fit2)
sum(x_val1$error^2)/nrow(a)

# Spherical
fit3 <- variofit(var1, cov.model="sph", ini.cov.pars=c(c1,alpha),
                 fix.nugget=FALSE, nugget=c0, weights = "cressie")
plot(var1); lines(fit3,lty=1)
x_val1 <- xvalid(b, model=fit3)
sum(x_val1$error^2)/nrow(a)

################################################
x.range <- (range(a$x))
x.range
y.range <- (range(a$y))
y.range
grd <- expand.grid(x=seq(from=x.range[1], to=x.range[2], by=0.005),
                   y=seq(from=y.range[1], to=y.range[2], by=0.005))

#Ordinary kriging predictions on the grid:
#Ordinary kriging using the krige.conv function:
qq <- krige.conv(b, locations=grd, krige=krige.control(obj.model=fit2))

#Construct a raster map:
#Use the image function:
```

Cynthia Lai

```
image(qq, main="Predicted values using OK")
points(a)

image(qq, values = qq$krige.var, main = "Variance of OK")
points(a)
```