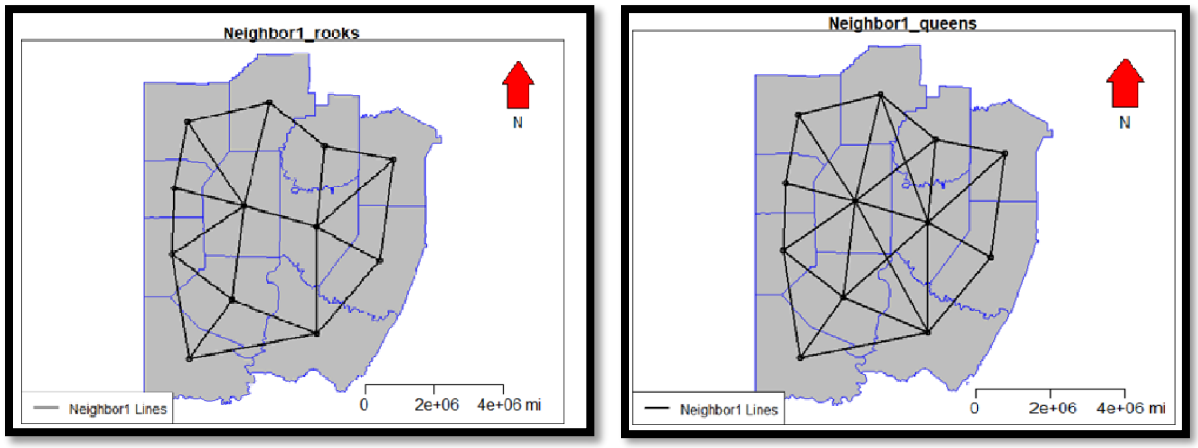
INFSCI 2809: Spatial Data Analytics
Spring 2019
Final Exam **Submitted by: Vaishnavi Deshpande**

PART A

Perform autocorrelation on the population attribute (POP_ARR02) in "1_Neighbor1.shp" and "1_Neighbor2.shp" files by using Rook's adjacency and Queen's adjacency. Use both Moran's I and Geary's C to measure the level, type, significance of correlation in the attribute.

Map for Neighbor1



Neighbor1 Moran's results for PartA.R

Method	Autocorrelation Result	p-value	Variance
Moran's rooks	0.08978452	0.1429	0.02866555
Moran's queens	0.02826381	0.1977	0.01966619

Results for individual files:[I tried the code given as solutions to Project4]

Method	Autocorrelation	p-value	Variance
Moran's rooks	0.1677115	0.06669662	0.02969266
Moran's I queen	0.1776717	0.03942496	0.02335660
Geary's C rooks	0.74402547	0.06622	0.02894427
Geary's C queens	0.72082742	0.04701	0.02779228

Moran's I value in both rooks an queens adjacency matrix is greater than 0 which indicates a positive autocorrelation. That implies data are positively correlated i.e. most pairs of the adjacent locations have the values on the same side of the mean and Moran's I has a positive value.
Thus, positive value indicates positive autocorrelation. Another point to be noted is that it's not strongly correlated because the index score is not 0.3 or more or -0.3 or less.

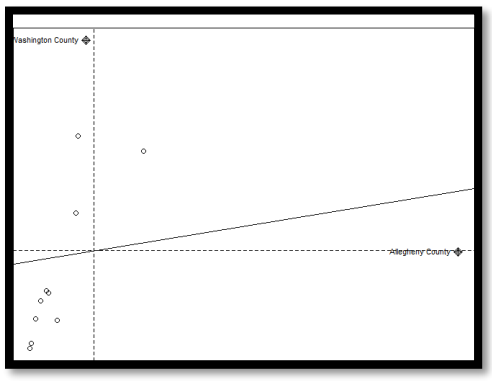
Moran's I is unlike most other correlation coefficients in that we can't take the index at face value. It is an inferential statistic. And we have to determine statistical significance before we can read the result. This is done with a simple hypothesis test, calculating its associated p-value.
The outcome of the hypothesis test is the P-value that leads us to either:

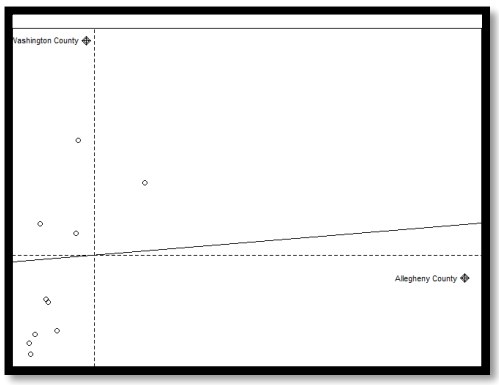
• Reject the null hypothesis if the p-value is low. We can then go on to conclude that our pattern is unlikely to have been produced by the hypothesized process; or
• Fail to reject the null hypothesis if the p-value is high. In this case, we conclude that there is insufficient evidence to believe that the observed pattern is not the outcome of the hypothesized process.
P-value is 0.066and 0.039 for rook's and queen's adjacency matrix for Neighbor1 indicating that it is statistically non-significant.

For Geary's C, for both rook's and queen's adjacency matrix the value of 0.74 and 0.72 which ranges between 0 and 1 and hence indicates positive autocorrelation.

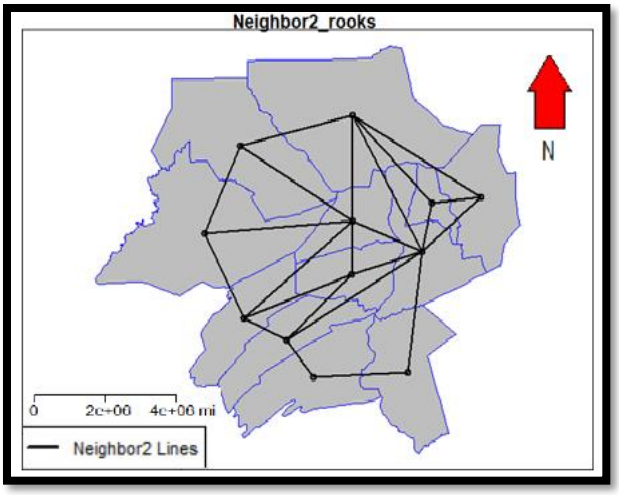
Based on the results it's clear that these polygons satisfy in terms of **independence, randomness, Tobler's law, Moran's I, and Geary's C.**

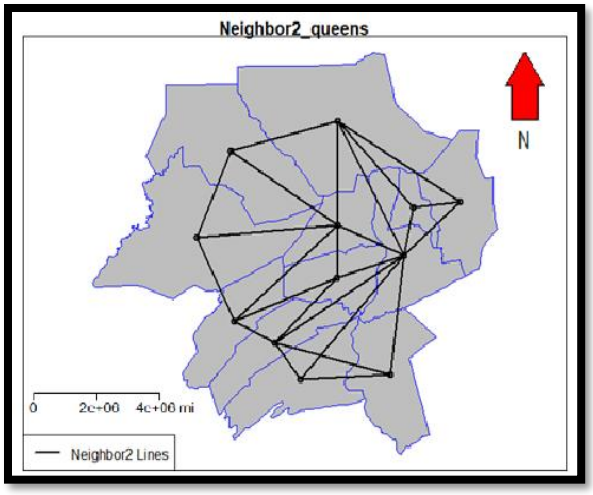
Below are the plots of Moran's I for the **rooks and the queen's adjacency respectively** based and these plots help us understand the **regression statistics** to verify our results of autocorrelation discussed above.





For Neighbor-2





Neighbor2 Results for PartA.R

Method	Autocorrelation Result	p-value	Variance
Moran's rooks	-0.07175519	0.4513	0.02445329
Moran's queens	-0.1635163	0.6905	0.02130757

Results for the individual files:

Method	Autocorrelation	p-value	Variance
Moran's I rooks	-0.01938815	0.3406	0.03032394
Moran's I queen	-0.02239824	0.3321	0.02491814
Geary's C rooks	0.9384975	0.3676	0.03307895
Geary's C queens	0.9369127	0.3578	0.02999825

Moran's I value in both rooks an queens adjacency matrix is less than 0 which indicates a negative autocorrelation. That implies data are points are more negatively correlated i.e. most pairs of the adjacent locations have the values on the other side of the mean and hence Moran's I has a negative value.
Thus, negative value indicates negative autocorrelation. Another point to be noted is that it's not strongly correlated because the index score is not 0.3 or more or -0.3 or less.

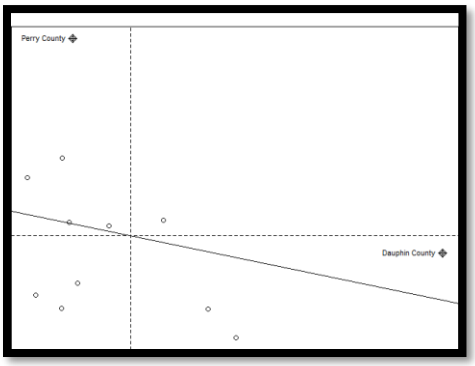
Moran's I is unlike most other correlation coefficients in that we can't take the index at face value. It is an inferential statistic. And we have to determine statistical significance before we can read the result. This is done with a simple hypothesis test, calculating its associated p-value.
The outcome of the hypothesis test is the P-value that leads us to either:

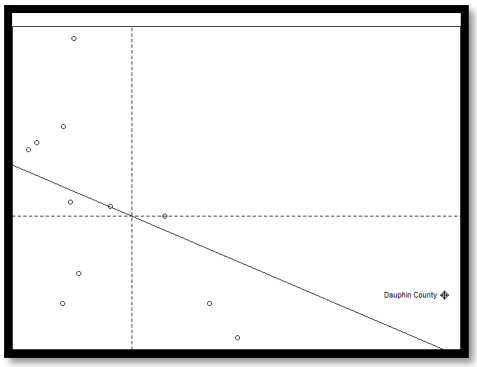
• Reject the null hypothesis if the p-value is low. We can then go on to conclude that our pattern is unlikely to have been produced by the hypothesized process; or
• Fail to reject the null hypothesis if the p-value is high. In this case, we conclude that there is insufficient evidence to believe that the observed pattern is not the outcome of the hypothesized process.
P-value as mentioned in the table for both rook's and queen's adjacency matrix for Neighbor2 also indicate that it is statistically non-significant.

For Geary's C, for both rook's and queen's adjacency matrix the value of 0.93849~>1 and 0.9369 ~>1 indicates negative autocorrelation.

Based on the results it's clear that these polygons satisfy in terms of **independence, randomness, Tobler's law, Moran's I, and Geary's C.**

Below are the plots of Moran's I for **the rooks and the queen's adjacency matrix respectively** based and these plots help us understand the **regression statistics** to verify our results of autocorrelation discussed above.





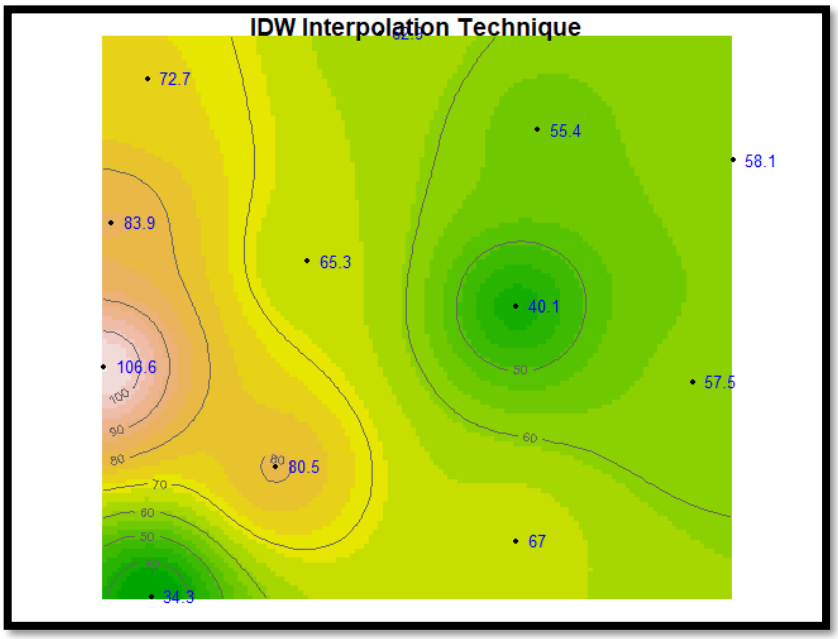
PART B

Use the Inverse Distance Weighting (IDW) and the Ordinary Kriging (OK) methods to interpolate an "Intensity" value at the center location of each polygon in "2_Community.shp" file using the sample "Intensity" values at locations (s_lat, s_long) in the same file. For IDW, use different values for r (window size) and different values for k (exponent of distance) to find the most similar map as produced by OK. Use exponential semiveriogram model for OK

Submit

• A table summarizing the IDW results for various values of r and k and the interpolated values of OK at the center locations of each polygon. [10 points]
• A map showing the IDW interpolated surface which is most similar to OK surface. [5 points]
• A map showing the OK interpolated surface. [5 points]
• A report discussing: [20 points] o The difference between the two surfaces, the one produced by IDW and the one produced by OK

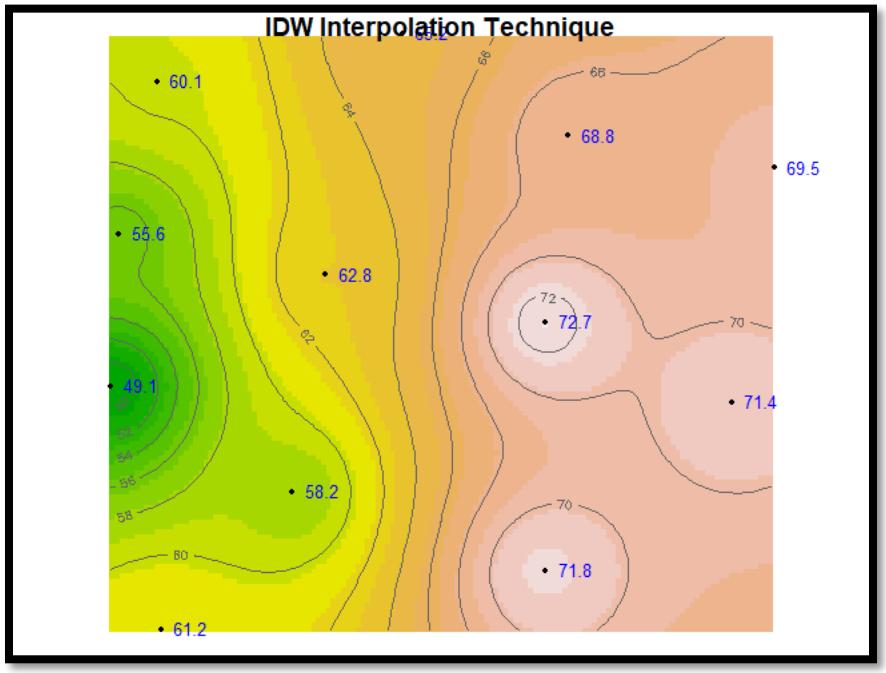
D=1



	x	y	z
0	-79.75728	41.40043	62.85763
1	-79.41957	41.19335	55.44142
2	-79.00053	41.12763	58.14239
3	-80.25668	41.30344	72.65515
4	-79.91287	40.91113	65.28912
5	-79.46588	40.81461	40.14555
6	-79.08792	40.65152	57.48537
7	-80.33295	40.99304	83.92601
8	-80.34930	40.68440	106.61875
9	-79.46670	40.31117	66.95852
10	-79.98108	40.46988	80.51255
11	-80.24858	40.19130	34.26385

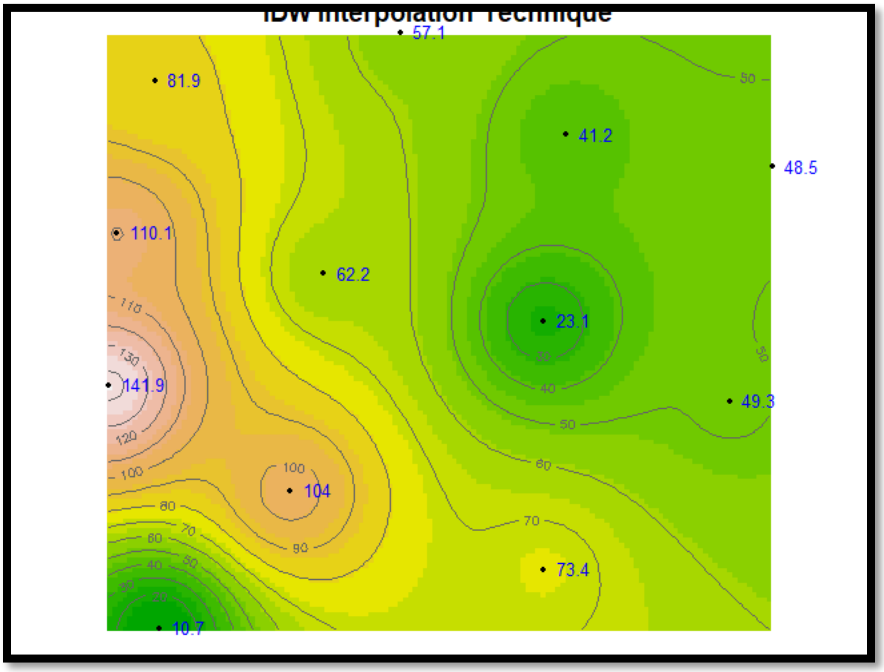
D=- 1 [Made the changes in distance=(distance) i.e. changed the power of distance]

	x	y	z
0	-79.75728	41.40043	65.18536
1	-79.41957	41.19335	68.80922
2	-79.00053	41.12763	69.48075
3	-80.25668	41.30344	60.12360
4	-79.91287	40.91113	62.77757
5	-79.46588	40.81461	72.72118
6	-79.08792	40.65152	71.35183
7	-80.33295	40.99304	55.56020
8	-80.34930	40.68440	49.09718
9	-79.46670	40.31117	71.77536
10	-79.98108	40.46988	58.22305
11	-80.24858	40.19130	61.20116



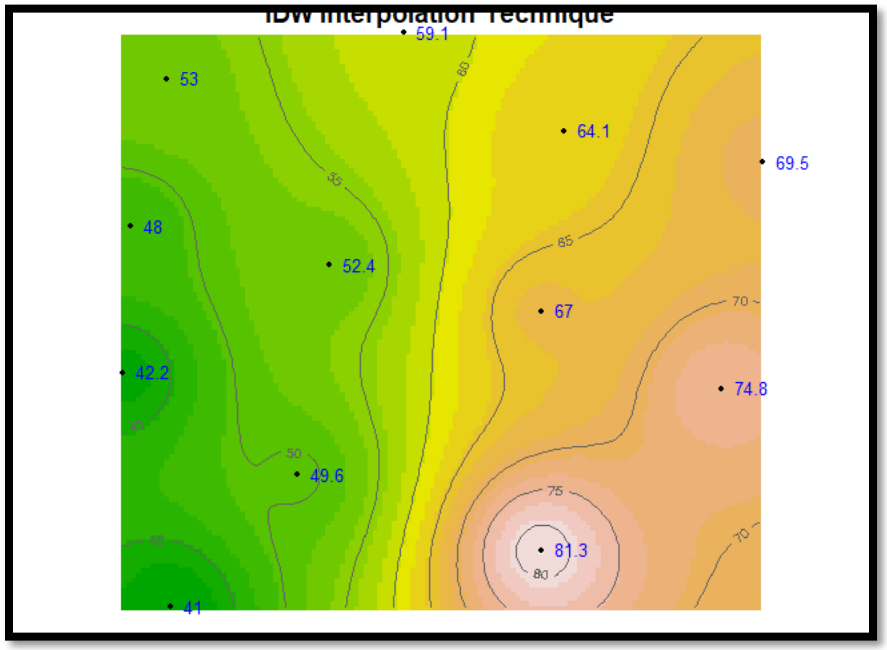
D=2

	x	y	z
0	-79.75728	41.40043	57.14900
1	-79.41957	41.19335	41.24478
2	-79.00053	41.12763	48.48932
3	-80.25668	41.30344	81.88386
4	-79.91287	40.91113	62.16784
5	-79.46588	40.81461	23.13641
6	-79.08792	40.65152	49.32386
7	-80.33295	40.99304	110.13397
8	-80.34930	40.68440	141.86349
9	-79.46670	40.31117	73.37536
10	-79.98108	40.46988	103.97854
11	-80.24858	40.19130	10.66706

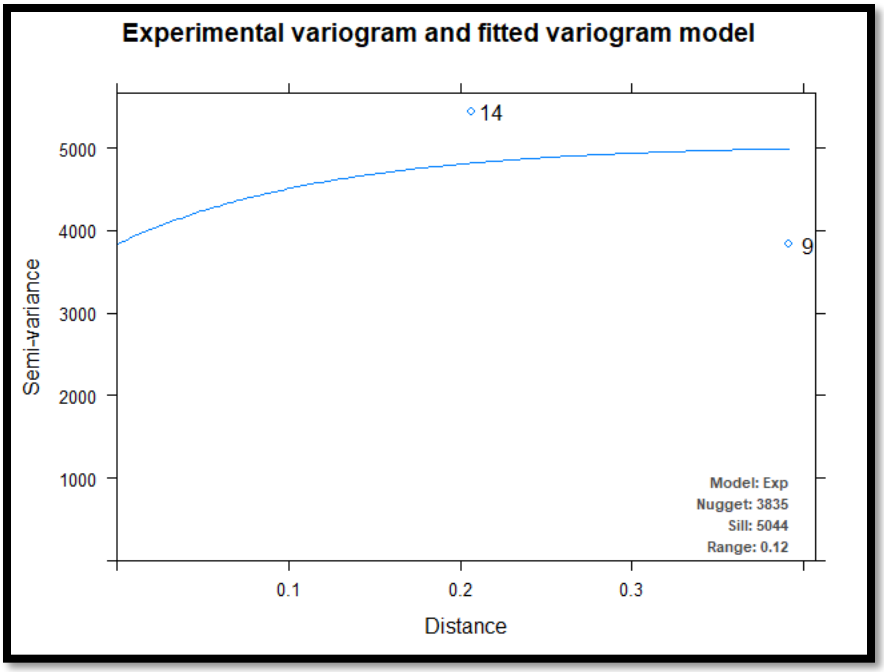


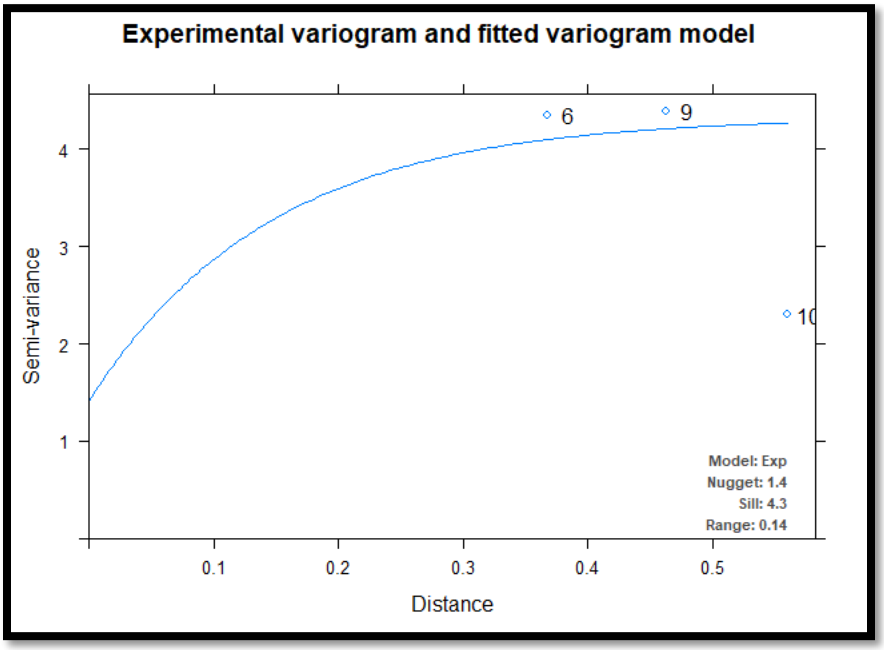
D=-3

	x	y	z
0	-79.75728	41.40043	59.12242
1	-79.41957	41.19335	64.12246
2	-79.00053	41.12763	69.45860
3	-80.25668	41.30344	52.98100
4	-79.91287	40.91113	52.37767
5	-79.46588	40.81461	66.97820
6	-79.08792	40.65152	74.83843
7	-80.33295	40.99304	48.00873
8	-80.34930	40.68440	42.15637
9	-79.46670	40.31117	81.33538
10	-79.98108	40.46988	49.61384
11	-80.24858	40.19130	40.96413

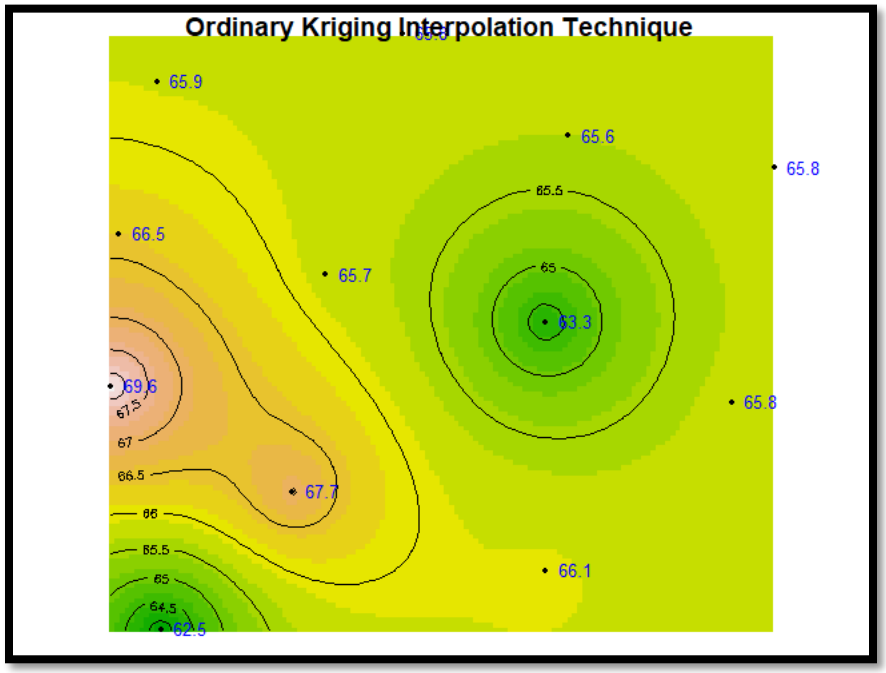


Ordinary Kriging results and plots


Experimental variogram and fitted variogram model
Semi-variance
Distance
14
9
Model: Exp
Nugget: 3835
Sill: 5044
Range: 0.12


Experimental variogram and fitted variogram model
Semi-variance
Distance
6
9
Model: Exp
Nugget: 1.4
Sill: 4.3
Range: 0.14

	x	y	z
13	-79.75728	41.40043	65.81488
14	-79.41957	41.19335	65.64915
15	-79.00053	41.12763	65.81893
16	-80.25668	41.30344	65.87400
17	-79.91287	40.91113	65.72861
18	-79.46588	40.81461	63.32197
19	-79.08792	40.65152	65.77734
20	-80.33295	40.99304	66.48884
21	-80.34930	40.68440	69.64460
22	-79.46670	40.31117	66.05502
23	-79.98108	40.46988	67.69970
24	-80.24858	40.19130	62.52180



Results and Analysis:

In general, the difference is,

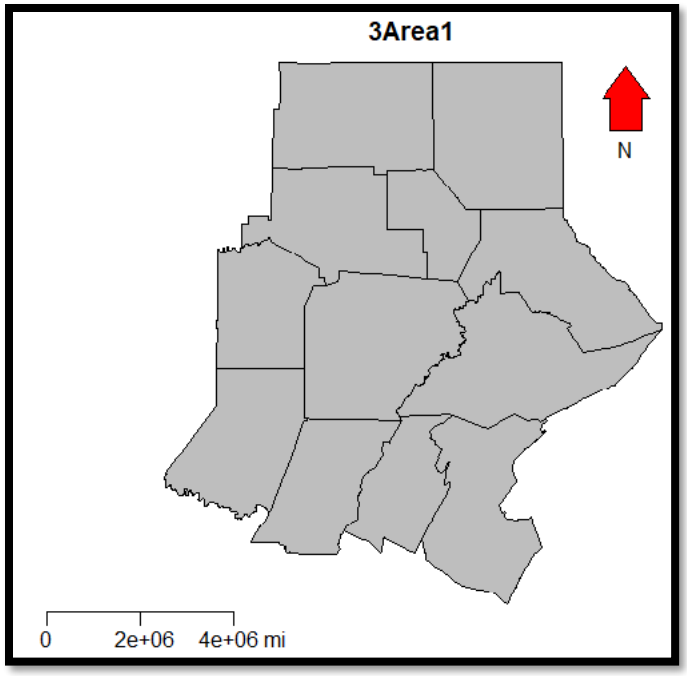
Inverse distance weighting (IDW), is a technique where the height of any continuous surface, zi, at location si is estimated as a distance-weighted sum of the sample values in some surrounding neighborhood.
Kriging is a statistical interpolation method that is optimal in the sense that it makes best use of what can be inferred about the spatial structure in the surface to be interpolated from an analysis of the control point data. Kriging is technique that use the control point data themselves to estimate the spatial structure in the underlying surface and use this information to determine appropriate spatial weights.
For our dataset, the results of IDW and OK technique are almost very close for d=-1. For IDW technique having d=-1,1,2,3 change the interpolation values not by very huge value.
Difference between the surfaces:
The surface produced using inverse distance-weighted interpolation with an inverse power law distance decay with k = -1. This surface shows some of the characteristic unrealistic "bulls-eye" effects that associates it with IDW interpolation.
The surface produced by ordinary kriging, with exponential semi variogram fitted to the data.
Although the estimates produced at any particular location do not differ much between these two maps, it is clear that the kriged surface appears much more realistic and that the method has some ability to adjust the spatial structure of estimates to reflect local variations in the surface structure.
From a theoretical perspective, there are two reasons to prefer kriging to simpler methods. First, if the correct model is used, the methods used in kriging have an advantage over other interpolation procedures in that the estimated values have minimum error associated with them. This is why the method is sometimes called optimum interpolation. Second, this error is quantifiable.
For every interpolated point an estimation variance can be calculated, which depends solely on the semi-variogram model, the spatial pattern of the points, and the calculated weights. The estimation variance is given by the weighted sum of the semi-variances of the distances from the control points to the location of the estimate.

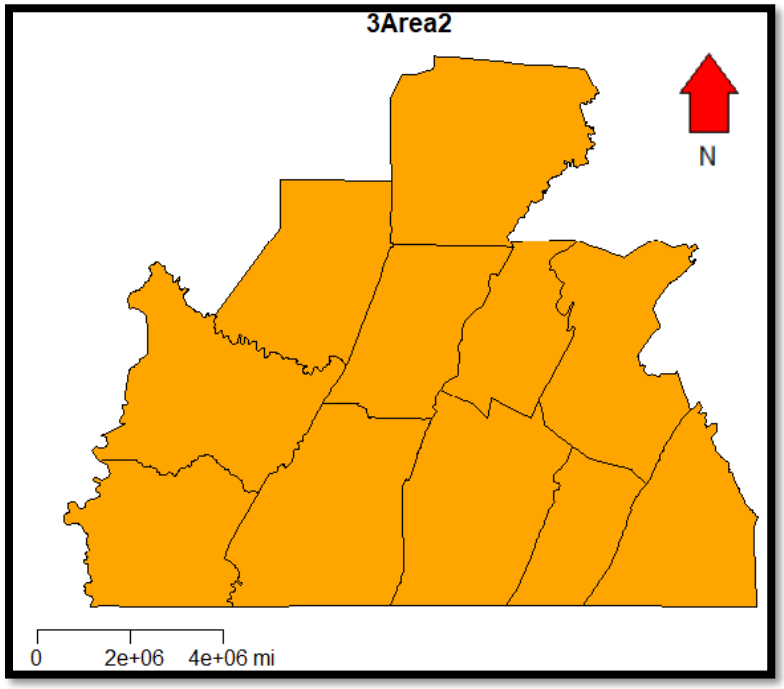
The values of d=-1 match with that of kriging technique. And when we compare the map for d=-1 with that of kriging, we notice that it is very close to OK technique. It might look different in this because its shifted to the left side. But the values in the table clearly suggests that it is close to the results obtained using the kriging technique.

Part C:

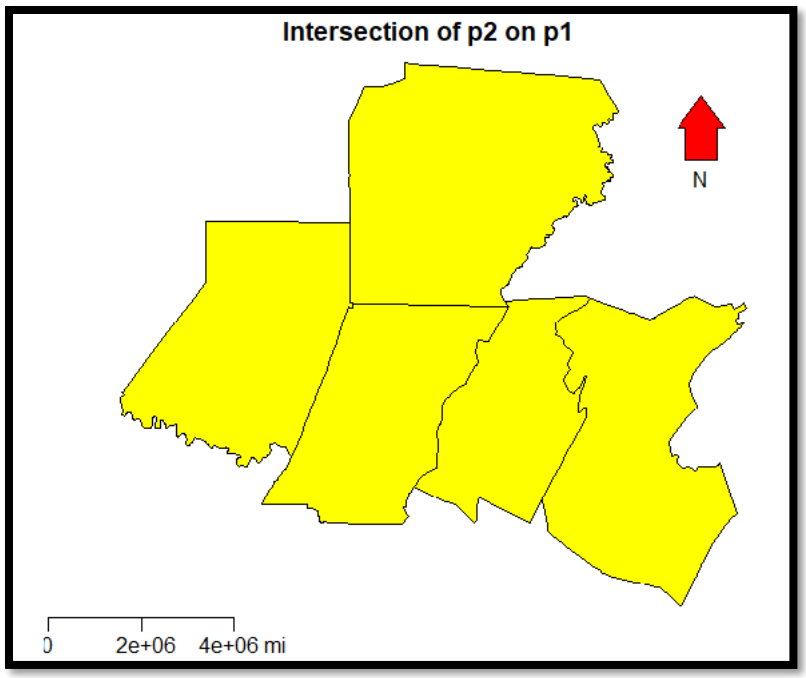
Obtain the following layers: P1 ("3_Area1.shp"), P2 ("3_Area2.shp"), P3 ("3_TrailPoints.shp"), and P4 ("3_State_Roads.shp") and perform the following:
a. Overlay P2 on P1 and find their intersection (P12), P1/ (= P1 - P12), and P2/ (= P2 - P12).
b. Overlay P3 on P1, P2, and P12 and find number of points within P1, P2, and P12.
c. Overlay P4 on P1, P2, and P12 and find the total length of road segments within P1, P2, and P12.

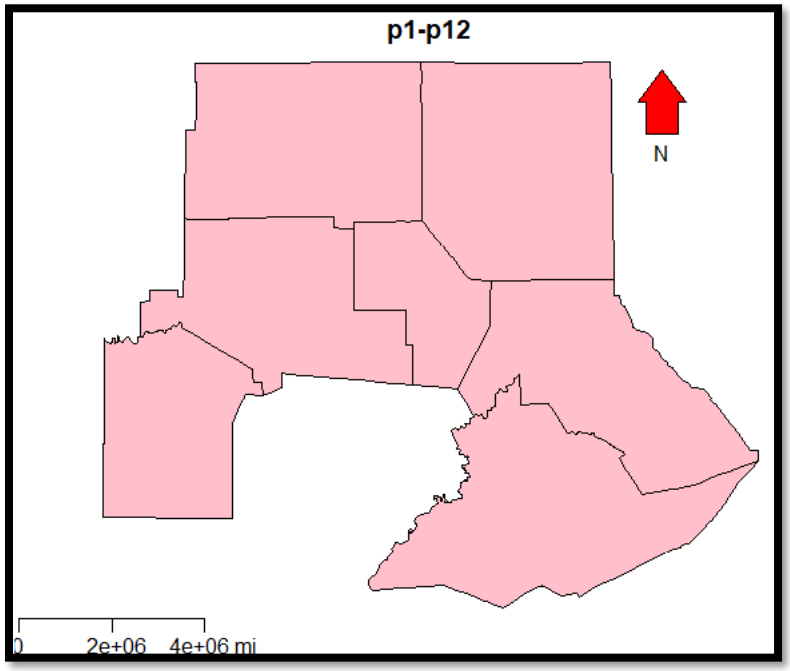
The maps of p1 and p2

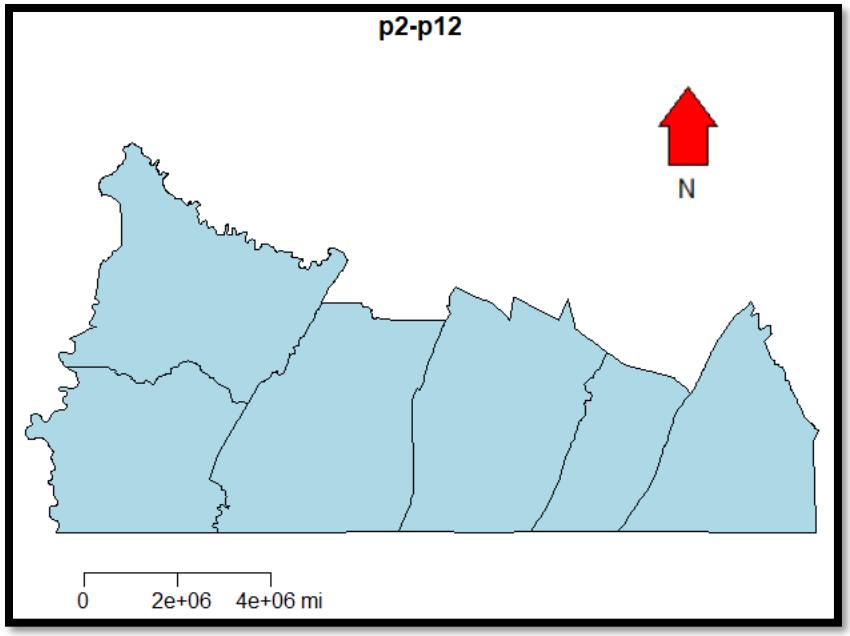




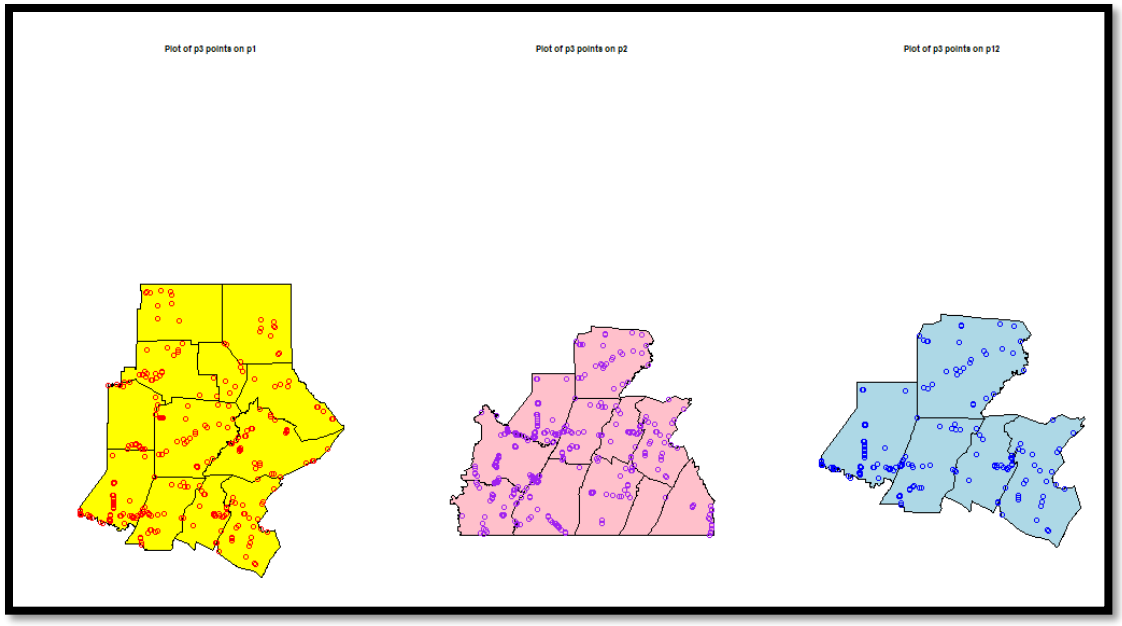
Maps of intersection of maps p1 and p2 , p1-p12 and p2-p12


Intersection of p2 on p1
N
0
2e+06
4e+06 mi

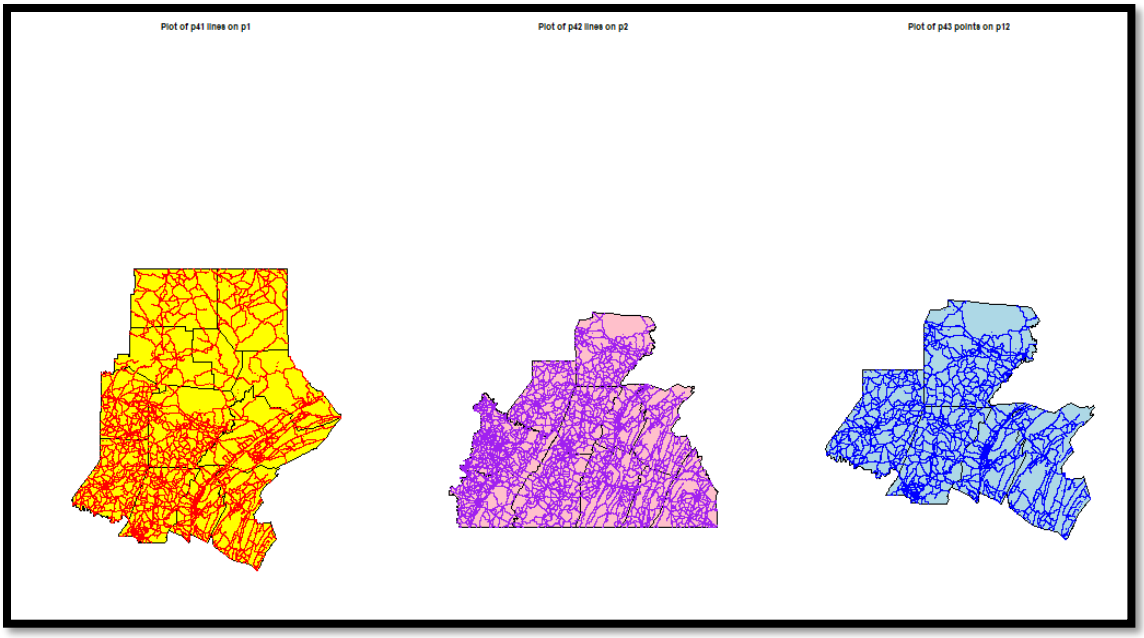

p1-p12
N
0
2e+06
4e+06 mi



Map showing the points within p1,p2 and p12.



Map showing the line of p4 on p1,p2 and p12.



Method and the results

Initially I started by looking at the individual maps,p1 and p2 are of type Spatial Polygon Dataframe .

So, in order to overlay I applied geometry function to convert them to Spatial polygons and then checked if the co-ordinate system are the same. P1 and P2 are both spatial polygons after applying the geometry and then overlaid p2 on p1 using over function. That helped me generate p1 intersect p2,p1-p12 and p2-p12.

Next, I looked at p3 and it Spatial Points Dataframe which I converted to Spatial points so that it is comfortable to work on spatial polygons.

p3 is in different CRS and hence I applied spTransform function to use the same CRS as p1 and p2. Then overlaid the points on polygon to find how many of the points are on p1 ,p2 and p12.

Next, I looked at p4 and it Spatial Lines Dataframe which I converted to Spatial Lines so that it is comfortable to work on spatial polygons.

p4 is in different CRS and hence I applied spTransform function to use the same CRS as p1 and p2. Then overlaid the lines on polygon to find how many of lines are on p1,p2 and p12.

Results:

Parameter	Value
Area of polygon p12:	1.132137
Number of points in p1	428
Number of points in p2	566

Number of points in p12	281
Number of lines in p1	15358
Number of lines in p2	20453
Number of lines in p12	8581

References:

https://rspatial.org/analysis/3-spauto.html :For Moran's I PartA.R results

https://rspatial.org/analysis/4-interpolation.html

https://courseweb.pitt.edu/bbcswebdav/pid-26197741-dt-content-rid-45904724_2/courses/2194_UPITT_INFSCI_2809_SEC1220/Project%204%20Solutions%202.pdf

https://rspatial.org/rosu/Chapter11.html

https://hautahi.com/rmaps

For part B I implemented the code given in the class.

Codes:

#PART A

PartA.R

```
#PART A
library(raster)
library(rgdal)#read the shape file
library(spatstat)
library(sp)
library(maptools)
library(ggplot2)
library(ggmap)
library(maps)
library(GISTools)
library(spdep)
```

```
setwd("C:/Users/rvais/Wdirectory/Finals")
getwd()


Neighbor1=readOGR("C:/Users/rvais/Wdirectory/Finals","1_Neighbor1")
Neighbor2=readOGR("C:/Users/rvais/Wdirectory/Finals", "1_Neighbor2")

#Representing shape file as Rook adjacency matrix
rooks = poly2nb(Neighbor1, row.names=Neighbor1$COUNTY, queen=FALSE)
class(rooks)
summary(rooks)

#Plotting the link between the polygons
library(GISTools)
par(mai=c(0,0,0.2,0))
plot(Neighbor1, col='gray', border='blue')
maps::map.scale(y = 360000, ratio=FALSE, relwidth=.2, metric=FALSE)
north.arrow(xb=1242280, yb=480342, len=5000, lab="N",col='Red')
legend('bottomleft', legend = 'Neighbor1 Lines', lty =1, lwd = 1, bty = 'o')
xy = coordinates(Neighbor1)
plot(rooks, xy, col='black', lwd=2, add=TRUE)
map.axes(cex.axis=0.8)
title('Neighbor1_rooks')


#Calculating Moran's I using rooks adjacency
n=length(Neighbor1@data$STATE)
wm = nb2mat(rooks, style='B')#A spatial weights matrix reflects the intensity of the geographic
relationship between observations
```

```
y = Neighbor1$POP_ARR02
ybar = mean(y)
dy = y - ybar
yi = rep(dy, each=n)
yj = rep(dy)
yiyj = yi * yj
pm = matrix(yiyj, ncol=n)#matrix of the multiplied pairs
pmw = pm * wm
spmw = sum(pmw)
smw = sum(wm)
sw  = spmw / smw
vr = n / sum(dy^2)
MI = vr * sw
MI
EI = -1/(n-1) ##Expected value of Moran's I
EI
#Create a 'listw' type spatial weights object to perform significance test
ww =  nb2listw(rooks, style='B')
moran(Neighbor1$POP_ARR02, ww, n=length(Neighbor1@data$STATE), S0=Szero(ww))
moran.test(Neighbor1$POP_ARR02, ww, randomisation=FALSE)
moran.mc(Neighbor1$POP_ARR02, ww, nsim=99)
rwm = mat2listw(wm, style='W')
mat = listw2mat(rwm)
apply(mat, 1, sum)[1:15]
moran.plot(y, rwm)

#queens for Neighbor1
queens = poly2nb(Neighbor1, row.names=Neighbor1$COUNTY, queen=TRUE)
class(queens)
```

```
summary(queens)
#Plotting the link between the polygons
par(mai=c(0,0,0.2,0))
plot(Neighbor1, col='gray', border='blue')
maps::map.scale(y = 360000, ratio=FALSE, relwidth=.2, metric=FALSE)
north.arrow(xb=1242280, yb=480342, len=5000, lab="N",col='Red')
map.axes(cex.axis=0.8)
title('Neighbor1_queens')
legend('bottomleft', legend = 'Neighbor1 Lines', lty = 1, lwd = 2, bty = 'o')
xy = coordinates(Neighbor1)
plot(queens, xy, col='black', lwd=2, add=TRUE)
#Calculating Moran's I using queen's adjacency
n=length(Neighbor1@data$STATE)
wm = nb2mat(queens, style='B')
y = Neighbor1$POP_ARR02
ybar = mean(y)
dy = y - ybar
yi = rep(dy, each=n)
yj = rep(dy)
yiyj = yi * yj
pm = matrix(yiyj, ncol=n)
pmw = pm * wm
spmw = sum(pmw)
smw = sum(wm)
sw  = spmw / smw
vr = n / sum(dy^2)
MI = vr * sw
MI
```

```
EI = -1/(n-1) ##Expected value of Moran's I
EI
#Create a 'listw' type spatial weights object to perform significance test
ww =  nb2listw(queens, style='B')
ww
moran(Neighbor1$POP_ARR02, ww, n=length(Neighbor1@data$STATE), S0=Szero(ww))
moran.test(Neighbor1$POP_ARR02, ww, randomisation=FALSE)
moran.mc(Neighbor1$POP_ARR02, ww, nsim=99)
rwm = mat2listw(wm, style='W')
mat = listw2mat(rwm)
apply(mat, 1, sum)[1:15]
moran.plot(y, rwm)

#For Neighbor 2
#Representing shape file as Rook adjacency matrix
rooks_neighbor2 = poly2nb(Neighbor2, row.names=Neighbor2$COUNTY, queen=FALSE)
class(rooks_neighbor2)
summary(rooks_neighbor2)

#Plotting the link between the polygons
par(mai=c(0,0,0.2,0))
graphics::plot(Neighbor2, col='gray', border='blue')
maps::map.scale(y = 360000, ratio=FALSE, relwidth=.2, metric=FALSE)
north.arrow(xb=1242280, yb=480342, len=5000, lab="N",col='Red')
map.axes(cex.axis=0.8)
title('Neighbor2_rooks ')
legend('bottomleft', legend = 'Neighbor2 Lines', lty = 1, lwd = 2, bty = 'o')
xy = coordinates(Neighbor2)
plot(rooks_neighbor2, xy, col='black', lwd=2, add=TRUE)
```

```
#Calculating Moran's I using rooks adjacency
n=length(Neighbor2@data$STATE)
wm = nb2mat(rooks_neighbor2, style='B')
n=length(Neighbor2@data$STATE)
y = Neighbor2$POP_ARR02
ybar = mean(y)
dy = y - ybar
yi = rep(dy, each=n)
yj = rep(dy)
yiyj = yi * yj
pm = matrix(yiyj, ncol=n)
pmw = pm * wm
spmw = sum(pmw)
smw = sum(wm)
sw  = spmw / smw
vr = n / sum(dy^2)
MI = vr * sw
MI
EI = -1/(n-1) ##Expected value of Moran's I
EI
#Create a 'listw' type spatial weights object to perform significance test
ww =  nb2listw(rooks_neighbor2, style='B')
moran(Neighbor2$POP_ARR02, ww, n=length(Neighbor2@data$STATE), S0=Szero(ww))
moran.test(Neighbor2$POP_ARR02, ww, randomisation=FALSE)
moran.mc(Neighbor2$POP_ARR02, ww, nsim=99)
rwm = mat2listw(wm, style='W')
mat = listw2mat(rwm)
apply(mat, 1, sum)[1:15]
```

```
moran.plot(y, rwm)


#queens for Neighbor2
queens_neighbor2 = poly2nb(Neighbor2, row.names=Neighbor2$COUNTY, queen=TRUE)
class(queens_neighbor2)
summary(queens_neighbor2)
#Plotting the link between the polygons
par(mai=c(0,0,0.2,0))
plot(Neighbor2, col='gray', border='blue')
maps::map.scale(y = 360000, ratio=FALSE, relwidth=.2, metric=FALSE)
north.arrow(xb=1242280, yb=480342, len=5000, lab="N",col='Red')
map.axes(cex.axis=0.8)
title('Neighbor2_queens')
legend('bottomleft', legend = 'Neighbor2 Lines', lty = 1, lwd = 2, bty = 'o')
xy = coordinates(Neighbor2)
plot(queens_neighbor2, xy, col='black', lwd=2, add=TRUE)
#Calculating Moran's I using queen's adjacency
n=length(Neighbor2@data$STATE)
wm = nb2mat(queens_neighbor2, style='B')
n=length(Neighbor2@data$STATE)
y = Neighbor2$POP_ARR02
ybar = mean(y)
dy = y - ybar
yi = rep(dy, each=n)
yj = rep(dy)
yiyj = yi * yj
pm = matrix(yiyj, ncol=n)
pmw = pm * wm
spmw = sum(pmw)
```

```
smw = sum(wm)
sw  = spmw / smw
vr = n / sum(dy^2)
MI = vr * sw
MI
EI = -1/(n-1) ##Expected value of Moran's I
EI
#Create a 'listw' type spatial weights object to perform significance test
ww =  nb2listw(queens_neighbor2, style='B')
ww
moran(Neighbor2$POP_ARR02, ww, n=length(Neighbor2@data$STATE), S0=Szero(ww))
moran.test(Neighbor2$POP_ARR02, ww, randomisation=FALSE)
moran.mc(Neighbor2$POP_ARR02, ww, nsim=99)
rwm = mat2listw(wm, style='W')
mat = listw2mat(rwm)
apply(mat, 1, sum)[1:15]
moran.plot(y, rwm)
```

Moran1neighbor.R

```
library(raster)
library(rgdal)#read the shape file
library(spatstat)
library(sp)
library(maptools)
library(ggplot2)
library(ggmap)
library(maps)
library(GISTools)
library(spdep)
```

```
moranXi<-function (values, Weights) {
 y <- values
 w <- Weights
 n <- length(y)
 m <- mean(y)
 sum <- rowSums(w)
 sum[sum == 0] <- 1
 w <- w/sum
 DL<-0;
 for(i in 1:n){
  DL<-DL+(y[i]-m)^2
 }
 I<-n/DL
 NR<-0
 DR<-0
 for(i in 1:n){
  for(j in 1:n){
   NR=NR+w[i,j]*(y[i]-m)*(y[j]-m)
   DR<-DR+w[i,j]
  }
 }
 I<-I*NR/DR
 I<-as.numeric(I)
 E_I_<- -1/(n-1)
 dy <- y-m
 S0 <- sum(w);
 S1 <- 0.5 * sum((w + t(w))^2)
 S2 <- sum((apply(w, 1, sum) + apply(w, 2, sum))^2)
```

```
 D <- (sum(dy^4))/(sum(dy^2))^2
 D <- D * n
 A <- n * ((n^2 - 3 * n + 3) * S1 - n * S2 + 3 * S0^2)
 B <- D * ((n^2 - n) * S1 - 2 * n * S2 + 6 * S0^2)
 C <- (n - 1) * (n - 2) * (n - 3) * S0^2
 E_I2_<- (A-B)/C
 E_I_2<- E_I_^2
 V_I_<- E_I2_-E_I_2
 z <- (I-E_I_)/sqrt(V_I_)
 p_value <- 1-pnorm(z)
 data.frame(moran=I,p_value=p_value)
}
#########################################################
#getArea() function : you should define a function for returning area of
#input polygon data
getArea<-function(PolygonData){
n<-length(PolygonData)
result<-vector();
for(i in 1:n){
v <-nrow(PolygonData@polygons[[i]]@Polygons[[1]]@coords)
area=0
for(j in 1:v){
if(j==v){
p=1
}else{
p=j+1
}
x1<-PolygonData@polygons[[i]]@Polygons[[1]]@coords[j,1]
y1<-PolygonData@polygons[[i]]@Polygons[[1]]@coords[j,2]
```

```
x2<-PolygonData@polygons[[i]]@Polygons[[1]]@coords[p,1]
y2<-PolygonData@polygons[[i]]@Polygons[[1]]@coords[p,2]
area<-area+(x1*y2-y1*x2)/2
}
result[i]<-abs(area)
}
return(result)
}
# Computing Area Vector for a shapefile type of polygon named X:
X <- readOGR("C:/Users/rvais/Wdirectory/Finals", "1_Neighbor1")
#calculate AlleghenyCounty_Municipal's Moran I
X.area <- getArea(X)
#calculateMoran'I and Geary C for polygon X
X.queen <- poly2nb(X,queen=TRUE)
X.rook <- poly2nb(X,queen=FALSE)
X.matrix.queen <- nb2mat(X.queen)
X.matrix.rook <- nb2mat(X.rook)
mi.q <- moranXi(X.area,X.matrix.queen)
mi.q
mi.r <-moranXi(X.area,X.matrix.rook)
mi.r
#Moran'I and Geary's C using R package
X.nb2listw.queen <- nb2listw(X.queen)
X.nb2listw.rook <- nb2listw(X.rook)
moran.test(X.area,X.nb2listw.queen)
moran.test(X.area,X.nb2listw.rook)
```

Moran2neighbor.R

```
library(raster)
library(rgdal)#read the shape file
library(spatstat)
library(sp)
library(maptools)
library(ggplot2)
library(ggmap)
library(maps)
library(GISTools)
library(spdep)

moranXi<-function (values, Weights) {
 y <- values
 w <- Weights
 n <- length(y)
 m <- mean(y)
 sum <- rowSums(w)
 sum[sum == 0] <- 1
 w <- w/sum
 DL<-0;
 for(i in 1:n){
  DL<-DL+(y[i]-m)^2
 }
 I<-n/DL
 NR<-0
 DR<-0
 for(i in 1:n){
  for(j in 1:n){
   NR=NR+w[i,j]*(y[i]-m)*(y[j]-m)
```

```
    DR<-DR+w[i,j]
   }
  }
  I<-I*NR/DR
  I<-as.numeric(I)
  E_I_<- -1/(n-1)
  dy <- y-m
  S0 <- sum(w);
  S1 <- 0.5 * sum((w + t(w))^2)
  S2 <- sum((apply(w, 1, sum) + apply(w, 2, sum))^2)
  D <- (sum(dy^4))/(sum(dy^2))^2
  D <- D * n
  A <- n * ((n^2 - 3 * n + 3) * S1 - n * S2 + 3 * S0^2)
  B <- D * ((n^2 - n) * S1 - 2 * n * S2 + 6 * S0^2)
  C <- (n - 1) * (n - 2) * (n - 3) * S0^2
  E_I2_<- (A-B)/C
  E_I_2<- E_I_^2
  V_I_<- E_I2_-E_I_2
  z <- (I-E_I_)/sqrt(V_I_)
  p_value <- 1-pnorm(z)
  data.frame(moran=I,p_value=p_value)
}
##########################################################
#getArea() function : you should define a function for returning area of
#input polygon data
getArea<-function(PolygonData){
n<-length(PolygonData)
result<-vector();
for(i in 1:n){
```

```
v <-nrow(PolygonData@polygons[[i]]@Polygons[[1]]@coords)
area=0
for(j in 1:v){
if(j==v){
p=1
}else{
p=j+1
}
x1<-PolygonData@polygons[[i]]@Polygons[[1]]@coords[j,1]
y1<-PolygonData@polygons[[i]]@Polygons[[1]]@coords[j,2]
x2<-PolygonData@polygons[[i]]@Polygons[[1]]@coords[p,1]
y2<-PolygonData@polygons[[i]]@Polygons[[1]]@coords[p,2]
area<-area+(x1*y2-y1*x2)/2
}
result[i]<-abs(area)
}
return(result)
}
# Computing Area Vector for a shapefile type of polygon named X:
X <- readOGR("C:/Users/rvais/Wdirectory/Finals", "1_Neighbor1")
#calculate AlleghenyCounty_Municipal's Moran I
X.area <- getArea(X)
#calculateMoran'I and Geary C for polygon X
X.queen <- poly2nb(X,queen=TRUE)
X.rook <- poly2nb(X,queen=FALSE)
X.matrix.queen <- nb2mat(X.queen)
X.matrix.rook <- nb2mat(X.rook)
mi.q <- moranXi(X.area,X.matrix.queen)
mi.q
```

```
mi.r <-moranXi(X.area,X.matrix.rook)
mi.r
#Moran'I and Geary's C using R package
X.nb2listw.queen <- nb2listw(X.queen)
X.nb2listw.rook <- nb2listw(X.rook)
moran.test(X.area,X.nb2listw.queen)
moran.test(X.area,X.nb2listw.rook)
```

Geary’s C Neighbor1.R

```
library(raster)
library(rgdal)#read the shape file
library(spatstat)
library(sp)
library(maptools)
library(ggplot2)
library(ggmap)
library(maps)
library(GISTools)
library(spdep)

# Geary'C Index: You need to have a function for computing Geary'C Index,
gearyXc()
# This function takes a vector of values and a matrix of weights(W) measures
#spatial autocorrelation using Geary's C.
gearyXc <- function(values, Weights) {
  y <- values
  w <- Weights
  n <- length(y)
  m <- mean(y)
```

```
 DL <- 0;
 for(i in 1:n){
  DL <- DL + (y[i]-m)^2;
 }
 L <- (n-1)/DL
 NR <- 0;
 DR <- 0;
 for(i in 1:n){
  for(j in 1:n){
   NR <- NR + w[i,j]*(y[i]-y[j])^2
   DR <- DR + 2*w[i,j]
  }
 }
 C<-L * NR/DR
 return(C)
}
############################################################
#getArea() function : you should define a function for returning area of
#input polygon data
getArea<-function(PolygonData){
 n<-length(PolygonData)
 result<-vector();
 for(i in 1:n){
  v <-nrow(PolygonData@polygons[[i]]@Polygons[[1]]@coords)
  area=0
  for(j in 1:v){
   if(j==v){
    p=1
   }else{
```

```
      p=j+1
    }
    x1<-PolygonData@polygons[[i]]@Polygons[[1]]@coords[j,1]
    y1<-PolygonData@polygons[[i]]@Polygons[[1]]@coords[j,2]
    x2<-PolygonData@polygons[[i]]@Polygons[[1]]@coords[p,1]
    y2<-PolygonData@polygons[[i]]@Polygons[[1]]@coords[p,2]
    area<-area+(x1*y2-y1*x2)/2
  }
  result[i]<-abs(area)
 }
 return(result)
}
# Computing Area Vector for a shapefile type of polygon named X:
X <- readOGR("C:/Users/rvais/Wdirectory/Finals", "1_Neighbor1")


X.area <- getArea(X)
#Calculate Geary's C
X.queen <- poly2nb(X,queen=TRUE)
X.rook <- poly2nb(X,queen=FALSE)
X.matrix.queen <- nb2mat(X.queen)
X.matrix.rook <- nb2mat(X.rook)
queen2<-gearyXc(X.area,X.matrix.queen)
queen2
rook2<-gearyXc(X.area,X.matrix.rook)
rook2
#Moran'I and Geary's C using R package
X.nb2listw.queen <- nb2listw(X.queen)
X.nb2listw.rook <- nb2listw(X.rook)
```

```
geary.test(X.area,X.nb2listw.queen)
geary.test(X.area,X.nb2listw.rook)
```

Geary's C Neighbor2.R

```
library(raster)
library(rgdal)#read the shape file
library(spatstat)
library(sp)
library(maptools)
library(ggplot2)
library(ggmap)
library(maps)
library(GISTools)
library(spdep)

# Geary'C Index: You need to have a function for computing Geary'C Index,
gearyXc()
# This function takes a vector of values and a matrix of weights(W) measures
#spatial autocorrelation using Geary's C.
gearyXc <- function(values, Weights) {
  y <- values
  w <- Weights
  n <- length(y)
  m <- mean(y)
  DL <- 0;
  for(i in 1:n){
    DL <- DL + (y[i]-m)^2;
  }
  L <- (n-1)/DL
  NR <- 0;
```

```
 DR <- 0;
 for(i in 1:n){
  for(j in 1:n){
   NR <- NR + w[i,j]*(y[i]-y[j])^2
   DR <- DR + 2*w[i,j]
  }
 }
 C<-L * NR/DR
 return(C)
}
#########################################################
#getArea() function : you should define a function for returning area of
#input polygon data
getArea<-function(PolygonData){
 n<-length(PolygonData)
 result<-vector();
 for(i in 1:n){
  v <-nrow(PolygonData@polygons[[i]]@Polygons[[1]]@coords)
  area=0
  for(j in 1:v){
   if(j==v){
    p=1
   }else{
    p=j+1
   }
   x1<-PolygonData@polygons[[i]]@Polygons[[1]]@coords[j,1]
   y1<-PolygonData@polygons[[i]]@Polygons[[1]]@coords[j,2]
   x2<-PolygonData@polygons[[i]]@Polygons[[1]]@coords[p,1]
   y2<-PolygonData@polygons[[i]]@Polygons[[1]]@coords[p,2]
```

```
    area<-area+(x1*y2-y1*x2)/2
   }
   result[i]<-abs(area)
  }
  return(result)
}
# Computing Area Vector for a shapefile type of polygon named X:
X <- readOGR("C:/Users/rvais/Wdirectory/Finals", "1_Neighbor2")



X.area <- getArea(X)
#calculate Geary C for polygon X
X.queen <- poly2nb(X,queen=TRUE)
X.rook <- poly2nb(X,queen=FALSE)
X.matrix.queen <- nb2mat(X.queen)
X.matrix.rook <- nb2mat(X.rook)
queen2<-gearyXc(X.area,X.matrix.queen)
queen2
rook2<-gearyXc(X.area,X.matrix.rook)
rook2
#Moran'I and Geary's C using R package
X.nb2listw.queen <- nb2listw(X.queen)
X.nb2listw.rook <- nb2listw(X.rook)
geary.test(X.area,X.nb2listw.queen)
geary.test(X.area,X.nb2listw.rook)
```

community.R [Part -B]

```
#Part-B
library(readr)
```

```
library(ggplot2)
library(gstat)
library(rspatial)
library(sp)
library(nlme)
library(scales)
library(dismo)
library(raster)
library(rgdal)
library(phylin)
library(sp)
library(rgdal)
library(spatstat)
library(maptools)
library(GISTools)
library(plyr)
library(gstat)
library(automap)



#Loading the dataset
community=readOGR("C:/Users/rvais/Wdirectory/Finals", "2_Community")
plot(community)
x<-community$S_Long
y<-community$S_Lat
centroidX<-coordinates(community)[,1]
centroidY<-coordinates(community)[,2]
nx<-length(x)
ncenter<-length(centroidX)
```

```
Rslt<-list()
ctr<-data.frame()
for(j in 1:ncenter){
  data<-data.frame()
  for(i in 1:nx){
    distance<-sqrt((x[i]-centroidX[j])^2+(y[i]-centroidY[j])^2)
    distance<-as.numeric(distance)
    data<-
rbind(data,data.frame(x=x[i],y=y[i],distance=(distance)^(1),Intensity=community$Intensity[i],centroidX=
centroidX[j],centroidY=centroidY[j]))
  }
  names(data)[4]<-"z"
  idx<-order(data$distance)
  data<-data[idx,]
  inverse_distance<-1/data$distance
  inverse_distance
  weight<-(inverse_distance)/sum(inverse_distance)
  weight
  weight_z<-data$z*weight
  weight_z
  IDW.table<-cbind(data,data.frame("Inversedistance"=inverse_distance,weight=weight,"weighted
value"=weight_z))
  IDW.table
  cat('total z for each center',community@data$Intensity[j],':')
  z<-sum(IDW.table$weighted.value)
  print(z)
  ctr<-rbind(ctr,data.frame(x=centroidX[j],y=centroidY[j],z=z))
  Rslt[[j]]<-IDW.table
}
Rslt
```

```
ctr
View(ctr)
IDW<-as.data.frame(ctr)
coordinates(IDW) = ~x + y
x.range<-as.numeric(bbox(IDW)[1,])
y.range<-as.numeric(bbox(IDW)[2,])
grd <- expand.grid(x = seq(from = x.range[1], to =x.range[2], by = 0.01),
                   y = seq(from = y.range[1],to = y.range[2], by = 0.01)) #
#expand points to grid
coordinates(grd) <- ~x + y
gridded(grd) <- TRUE
dat.idw <- gstat::idw(ctr$z ~ 1, locations=IDW,newdata=grd)
OP<- par( mar=c(0,0,0,0))
image(dat.idw,"var1.pred",col=terrain.colors(20))
contour(dat.idw,"var1.pred", add=TRUE, nlevels=10, col="#656565")
plot(IDW, add=TRUE, pch=16, cex=0.5)
text(coordinates(community), as.character(round(ctr$z,1)), pos=4, cex=0.8,
    col="blue")
par(OP)
title(main="IDW Interpolation Technique")

#OK Interpolation
x<-community$S_Long
y<-community$S_Lat
centroidX<-coordinates(community)[,1]
centroidY<-coordinates(community)[,2]
nx<-length(x)
ncenter<-length(centroidX)
Rslt<-list()
```

```
ctr<-data.frame()
for(j in 1:ncenter){
  dataset<-data.frame()
  for(i in 1:nx){
    dataset<-rbind(dataset,data.frame(x=x[i],y=y[i],Intensity=community$Intensity[i]))
     }
  }
names(dataset)<-c("x","y","z")

community.length<-nrow(dataset)
centers.data<-data.frame(x=centroidX,y=centroidY,z=NA)
dataset<-rbind(dataset,centers.data)
rownames(dataset)<-1:nrow(dataset)
dataset
vgm1=dataset[1:community.length,]
coordinates(vgm1) = ~x + y
vgm1<-autofitVariogram(z~x+y,vgm1,model="Exp")
vgm1
plot(vgm1)
# function
gamma<-function(distance)
{
  result=14+(15-14)*(1-exp(-(abs(distance)/0.11)))
  return(result)
}
#calculate D
points<-nrow(dataset)
dist<-matrix(nrow = points+1,ncol =points+1)
dim(dist)
```

```
for(i in 1:points){
  for(j in 1:points){
    distance<-sqrt((dataset[i,1]-dataset[j,1])^2+(dataset[i,2]-
                                        dataset[j,2])^2)
    distance<-as.numeric(distance)
    dist[i,j]=distance
    dist[j,i]=distance
  }
}
A<-dist
A<-A[1:(community.length+1),1:(community.length+1)]
dim(A)
A
# functions, need discussing
gama.a<-gamma(A)
gama.a[community.length+1,]=c(rep(1,community.length),0)
gama.a[,community.length+1]=c(rep(1,community.length),0)
diag(gama.a)<-0
gama.a
idx<-which(is.na(dataset$z))#get the index whose value is NA
idx
for(k in idx){
  #calcualte b and d
  d<-dist[k,1:community.length]
  d
  # functions, need discussing
  b<-gamma(d)
  b[length(b)+1]=1
  b
```

```
  w<-solve(gama.a,b)
  z<-w[1:community.length]*dataset$z[1:community.length]
  z<-sum(z)
  dataset$z[k]<-z
}
dataset
cat('the centers\' info are')
result<-dataset[idx,]
result
#OK map
dat<-as.data.frame(result)
coordinates(dat) = ~x + y
vgm2<-autofitVariogram(z~x+y,dat,model="Exp")
vgm2
plot(vgm2)
nugget2=0.01
sill2=0.02
range2=0.17
m <- vgm(range=range2, "Exp", nugget=nugget2, psill=sill2)
dat.krg <- krige(dat$z~1, dat, grd, model = m)
OP<- par( mar=c(0,0,0,0))
image(dat.krg ,"var1.pred",col=terrain.colors(20))
contour(dat.krg ,"var1.pred", add=TRUE, nlevels=10)
plot(dat, add=TRUE, pch=16, cex=0.5)
text(coordinates(dat), as.character(round(dat$z,1)), pos=4, cex=0.8,
    col="blue")
par(OP)
title(main="Ordinary Kriging Interpolation Technique")
```

Part C

Partcoverlay.R

```
#PartC
library(raster)
library(rgdal)
library(phylin)
library(sp)
library(spatstat)
library(maptools)
library(GISTools)
library(plyr)
library(gstat)
library(automap)
library(rgeos)

p1=readOGR("C:/Users/rvais/Wdirectory/Finals", "3_Area1")
p2=readOGR("C:/Users/rvais/Wdirectory/Finals", "3_Area2")
p3=readOGR("C:/Users/rvais/Wdirectory/Finals", "3_TrailPoints")
p4=readOGR("C:/Users/rvais/Wdirectory/Finals", "3_State_Roads")


plot(p1,main="3Area1",col="grey",cex=0.8)
maps::map.scale(y = 360000, ratio=FALSE, relwidth=.2, metric=FALSE)
north.arrow(xb=1242280, yb=480342, len=5000, lab="N",col='Red')
plot(p2,main="3Area2",col="orange",cex=0.8)
maps::map.scale(y = 360000, ratio=FALSE, relwidth=.2, metric=FALSE)
north.arrow(xb=1242280, yb=480342, len=5000, lab="N",col='Red')
plot(p3,main="3_trailpoints",cex=0.8)
```

```
plot(p4,main="3_stateroads",cex=0.8)



#checking for the projetion of the 2 files to verify if theyare in the same co-ordinate system
p1=geometry(p1)
class(p1)
projection(p1)

p2=geometry(p2)
class(p2)
projection(p2)

####Yes,they belong to same CRS so we can overlay plot p2 on p1

##a
over(p2,p1)#Overlay p2 on p1

p12= intersect(p2, p1)
p12
sum(poly.areas(p12)) #1.132137
plot(p12,col="yellow",main="Intersection of p2 on p1 ")
maps::map.scale(y = 360000, ratio=FALSE, relwidth=.2, metric=FALSE)
north.arrow(xb=1242280, yb=480342, len=5000, lab="N",col='Red')
p11=p1-p12 #difference of plot1-plot12
plot(p11,col="pink",main="p1-p12")
maps::map.scale(y = 360000, ratio=FALSE, relwidth=.2, metric=FALSE)
north.arrow(xb=1242246, yb=480342, len=5000, lab="N",col='Red')
p22=p2-p12
plot(p22,col="light blue",main="p2-p12")
```

```
maps::map.scale(y = 360000, ratio=FALSE, relwidth=.2, metric=FALSE)

north.arrow(xb=1242280, yb=480342, len=5000, lab="N",col='Red')

##bMap showing points

p3=geometry(p3)

class(p3)

projection(p3)# projection is different from that of p1,p2, and p12


p12=geometry(p12)

class(p12)

projection(p12)

summary(p12)



library(rgdal)

TA=CRS("+proj=longlat +datum=NAD27 +no_defs +ellps=clrk66
+nadgrids=@conus,@alaska,@ntv2_0.gsb,@ntv1_can.dat")

p3TA=spTransform(p3,TA)

projection(p3TA)

over(p3TA,p1)

over(p3TA,p2)

over(p3TA,p12)

bbox(p3TA)

head(p3TA@coords)

p31= intersect(p3TA, p1)

p31#428

sum(poly.counts(p31,p1))

p32= intersect(p3TA, p2)

p32#566

sum(poly.counts(p32,p2))
```

```
p33= intersect(p3TA, p12)
p33#281
sum(poly.counts(p33,p2))
par(mfrow=c(1,3))
plot(p1,col="yellow"); points(p31,col="red"); title(main = list("Plot of p3 points on p1", cex=0.8))
plot(p2,col="pink"); points(p32,col="purple"); title(main = list("Plot of p3 points on p2", cex=0.8))
plot(p12,col="light blue"); points(p33,col="blue"); title(main = list("Plot of p3 points on p12", cex=0.8))

#Map showing the lines
p4=geometry(p4)
class(p4)
projection(p4)# projection is different from that of p1,p2, and p12
p4TA=spTransform(p4,TA)#Transforming to the Co-ordinate system of p1,p2,p12
projection(p4TA)
over(p4TA,p1)
over(p4TA,p2)
over(p4TA,p12)
bbox(p4TA)
head(p4TA@lines)
p41= intersect(p4TA, p1)
p41#15358
p42= intersect(p4TA, p2)
p42#20453
p43= intersect(p4TA, p12)
p43#8581
par(mfrow=c(1,3))
plot(p1,col="yellow"); lines(p41,col="red"); title(main = list("Plot of p41 lines on p1", cex=0.8))
plot(p2,col="pink"); lines(p42,col="purple"); title(main = list("Plot of p42 lines on p2", cex=0.8))
plot(p12,col="light blue"); lines(p43,col="blue"); title(main = list("Plot of p43 points on p12", cex=0.8))
```