Project 4

Jiawen Qi (jiq10)

April 19, 2017

PART A

Note: The dataset has some error in area, and I have fixed this problem.

*ACC: AlleghenyCounty_Council; ACM: AlleghenyCounty_Municipal

Shapefile	Method	Adjacency	I/C value	Expectation	Variance	Standard Deviate	p-value
ACC	Moran	Rook	0.31108273	-0.08333333	0.02560847	2.4647	0.006857
ACC	Moran	Queen	0.31108273	-0.08333333	0.02560847	2.4647	0.006857
ACC	Geary	Rook	0.6409275	1.0000000	0.0250958	2.2666	0.01171
ACC	Geary	Queen	0.6409275	1.0000000	0.0250958	2.2666	0.01171
ACM	Moran	Rook	-0.002915556	-0.007751938	0.003500366	0.081745	0.4674
ACM	Moran	Queen	-0.0001232697	-0.0077519380	0.0033378781	0.13204	0.4475
ACM	Geary	Rook	2.134989922	1.000000000	0.006785615	-13.778	1
ACM	Geary	Queen	2.09170840	1.00000000	0.00645514	-13.588	1

Interpretion:

For Allegheny County Council shapefile, no matter what method I use, the rook's result is always the same is queen's result. Rook's adjacency is sharing the same edge. Queen's adjacency is sharing same corner. When the results are same, it means there is no neighbor at the corner. When using Moran's method, the I > 0.3, which means a strong positive autocorrelation, and the p-value < 0.05 which means the result is strong trustable. When using Geary's method, the 0<C<1 indicates a positive autocorrelation, and the p-value tells me this result is trustable, too.

For Alleghney County Municipal shapefile, the results for rook's and queen's are different, which means there are some neighbors at the corner. When using Moran's method, although it is negative, it is so close to 0 and the p-value is >0.05, which means, there is a very very very week negative autocorrelation, it is even can be treat as no autocorrelation in it, and this result is very week to trust. When using Geary's method, the p-value is even equals to 1, which means we don't need to see the result, it's totally untrustable.

PART B

Global G Statistic Result:

Global G Statistic	Expectation	Variance	Standard Deviate	P-value
2.974714e-02	1.515152e-02	1.600546e-05	3.6483	0.000132

Interpretion: 2.973714e-02 > 1.515152e-02, when global G statistics is greater than expected value, which means high values clustered together, and it's a kind of "hot pots" potentially. The p-value makes us trust this result.

Geographically Weighted Regression (GWR):

Summary of GWR coefficient estimates at data points						
	Min.	**1st Qu.**	**Median**	**3rd Qu.**	**Max.**	**Global**
X.Intercept.	1.070e+01	2.533e+02	4.138e+02	5.149e+02	7.124e+02	427.8579
POP_CRI01	3.778e-02	4.646e-02	5.528e-02	5.780e-02	5.972e-02	0.0575
AG_CRI01	-4.415e+02	-3.710e+02	-3.038e+02	-1.634e+02	-5.376e+01	-333.0392

Area	-5.774e+03	-4.211e+03	-3.389e+03	-2.002e+03	-5.955e+02	-3255.5642

The equation:

$$\widehat{Index}01 = (10.1 \text{ to } 712.4) + (0.03778\ to\ 0.05972)POPCRI01 + (-441.5\ to - 537.6)AGCRI01 + (-5774\ to - 595.5)Area$$

Global equation:
$$\widehat{Index}01 = 427.8579 + 0.0575 * POP\_CRI01 - 333.0392 * AG\_CRI01 - 3255.5642 * Area$$

The original value of Index01 for Mifflin county is: 215

The prediction result of Mifflin county is: 105.36254

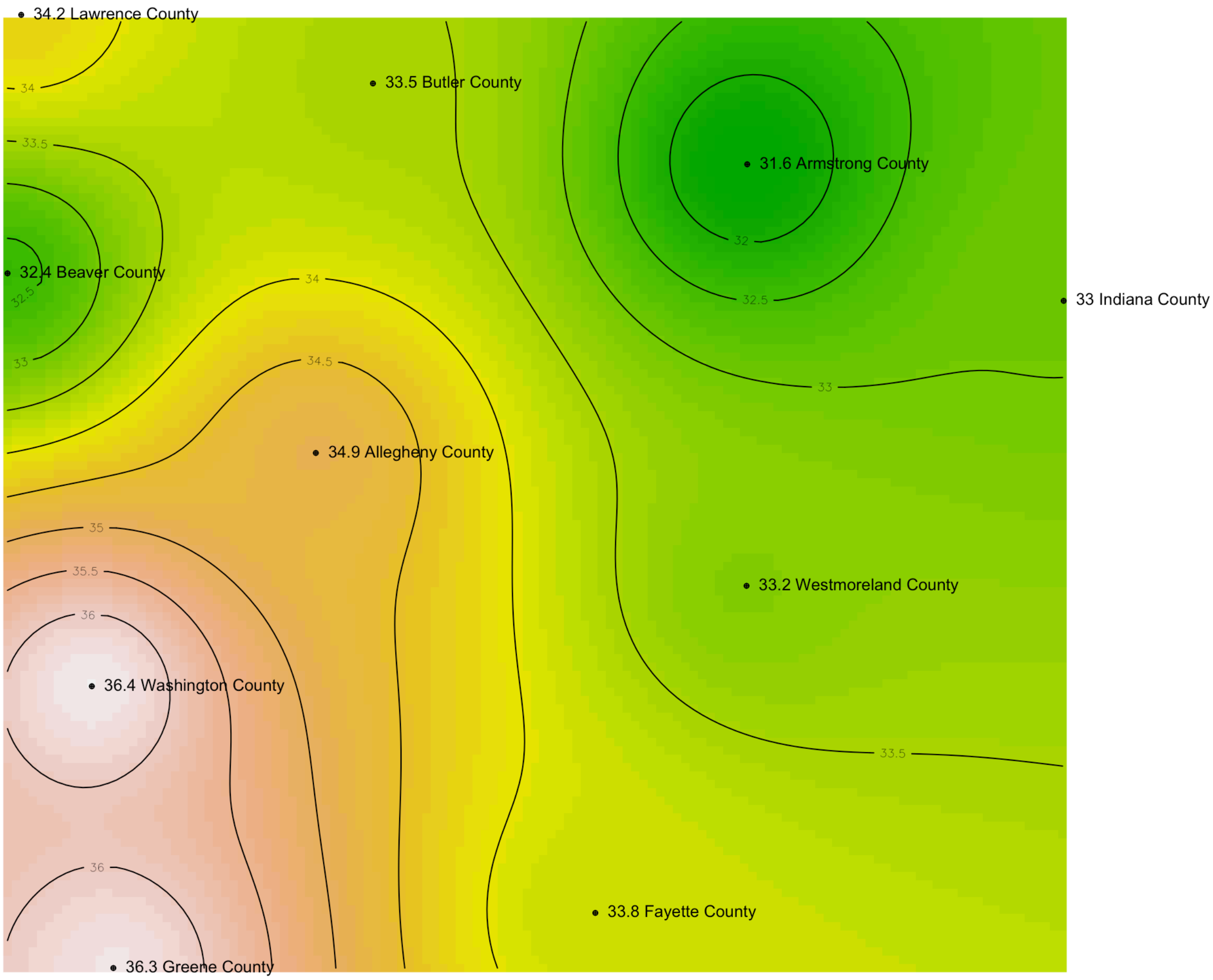
PART C

Using IDW techinique:

IDW interpolated results			
County	**x**	**y**	**z (Interpolated value)**
Butler	-79.91287	40.91113	33.53768
Armstrong	-79.46588	40.81461	31.62437
Indiana	-79.08792	40.65152	32.97690
Lawrence	-80.33295	40.99304	34.24218
Beaver	-80.34930	40.68440	32.39437
Westmoreland	-79.46670	40.31117	33.16840
Allegheny	-79.98108	40.46988	34.85891

Washington	-80.24858	40.19130	36.40584
Fayette	-79.64728	39.92059	33.78217
Greene	-80.22289	39.85479	36.32539

The Map using IDW:

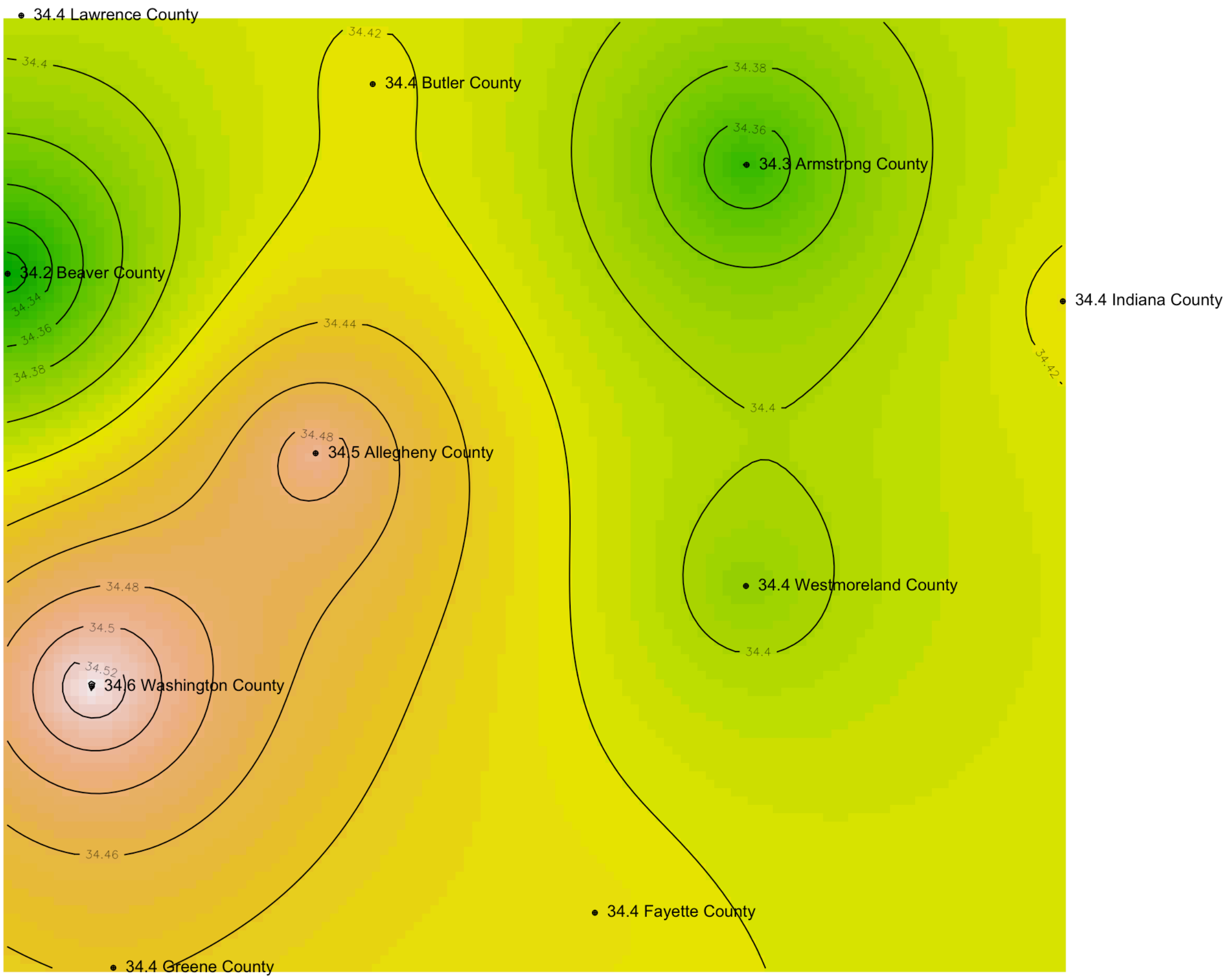


Using OK technique:

OK interpolated results			
County	**x**	**y**	**z (Interpolated value)**
Butler	-79.91287	40.91113	34.42886

Armstrong	-79.46588	40.81461	34.29668
Indiana	-79.08792	40.65152	34.43313
Lawrence	-80.33295	40.99304	34.41276
Beaver	-80.34930	40.68440	34.23777
Westmoreland	-79.46670	40.31117	34.36798
Allegheny	-79.98108	40.46988	34.53589
Washington	-80.24858	40.19130	34.60604
Fayette	-79.64728	39.92059	34.43163
Greene	-80.22289	39.85479	34.44560

The Map Using OK:



The differences between two techniques:

For IDW, nearer locations are given more prominence in calculating the local mean. It used known z values and weights determined as a function of distances between the unknown and known points. IDW differs from Kriging because there is no statistical models used in it. There is no determination of spatial autocorrelation taken into considereation. IDW has the advantage that it is easy to define and understand the results. Kriging is most appropriate when you know there is a spatially correlated distance or directional bias in the data. Kriging is a statistical method that makes use of a variogram to calculate the spatial autocorrelation. The weights in Kriging are helped determined by the semivariogram.

The differences between two maps:

The OK map looks more detailed, smooth, and “natural” than IDW map.

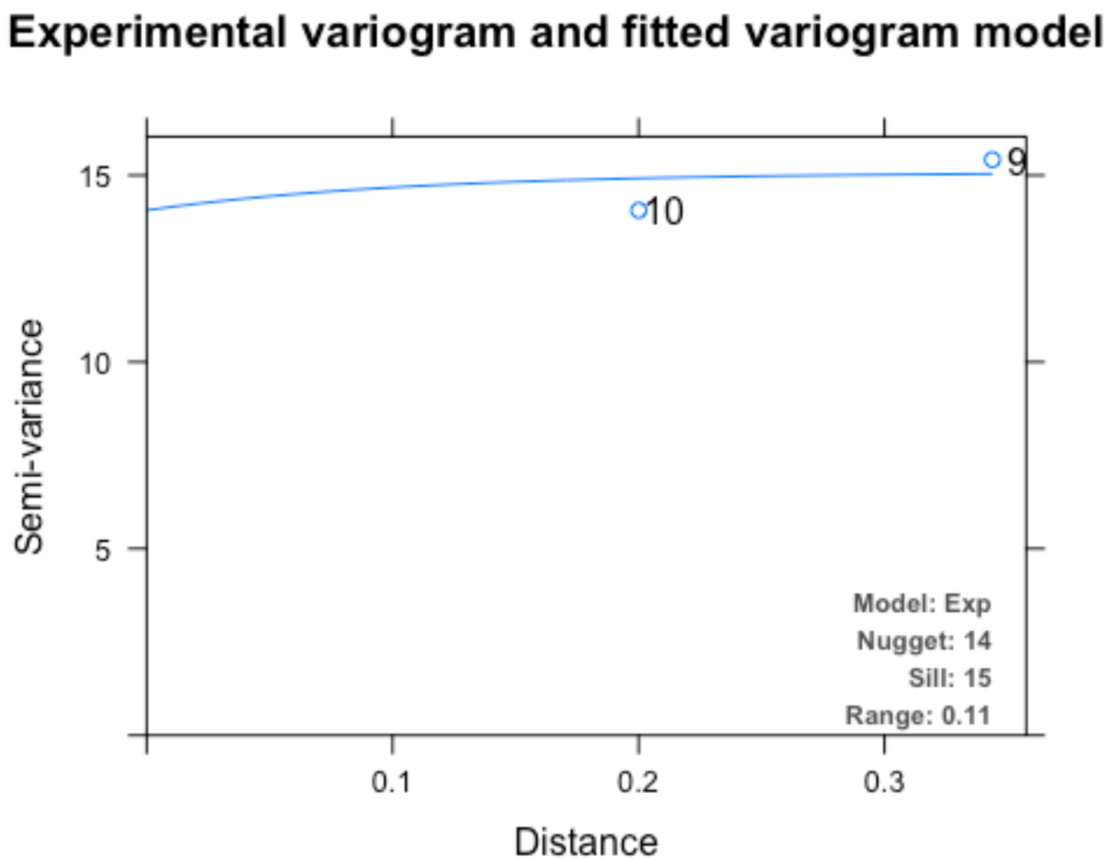
Code

```
##### Load Libraries #####
library(rgdal)
library(UScensus2010)
library(spdep)
library(GWmodel)
library(spgwr)
library(dplyr)
library(fields)
library(gstat)
library(automap)

##### PART A #####
#Import files
AlleghenyCountyCouncil <- readOGR(".", "AlleghenyCounty_Council")  # import A
lleghenyCounty_Council shapefile
AlleghenyCountyMunicipal <- readOGR(".", "AlleghenyCounty_Municipal")  # impo
rt AlleghenyCounty_Municipal shapefile

# ACC is the abbr of AlleghenyCountyCouncil; ACM is the abbr of
# AlleghenyCountyMunicipal
ACC.area = areaPoly(AlleghenyCountyCouncil)  # calculate the area of each pol
ygon in ACC
ACM.area = areaPoly(AlleghenyCountyMunicipal)  # calculate the area of each p
olygon in ACM

## construct neighbours list from polygon
ACC.queen <- poly2nb(AlleghenyCountyCouncil, queen = TRUE)
ACC.rook <- poly2nb(AlleghenyCountyCouncil, queen = FALSE)
ACM.queen <- poly2nb(AlleghenyCountyMunicipal, queen = TRUE)
ACM.rook <- poly2nb(AlleghenyCountyMunicipal, queen = FALSE)

# change neighbour list to listw (get the spatial weights for neighbours
# lists)
ACC.nb2listw.queen <- nb2listw(ACC.queen)
ACC.nb2listw.rook <- nb2listw(ACC.rook)
ACM.nb2listw.queen <- nb2listw(ACM.queen)
ACM.nb2listw.rook <- nb2listw(ACM.rook)

# calculate moran's and geary'c
moran.test(ACC.area, ACC.nb2listw.queen, randomisation = FALSE)  # ACC moran
queen: 0.31108273, Expectation = -0.08333333
moran.test(ACC.area, ACC.nb2listw.rook, randomisation = FALSE)  # ACC moran r
ook: 0.31108273, Expectation = -0.08333333
geary.test(ACC.area, ACC.nb2listw.queen, randomisation = FALSE)  # ACC geary
queen: 0.6409275, Expectation = 1;
geary.test(ACC.area, ACC.nb2listw.rook, randomisation = FALSE)  # ACC geary r
ook: 0.6409275, Expectation = 1;
moran.test(ACM.area, ACM.nb2listw.queen, randomisation = FALSE)  # ACM moran
```

```r
queen: -0.0001232697, Expectation = -0.007751938;
moran.test(ACM.area, ACM.nb2listw.rook, randomisation = FALSE)  # ACM moran r
ook: -0.002915556, Expectation = -0.007751938
geary.test(ACM.area, ACM.nb2listw.queen, randomisation = FALSE)  # ACM geary
queen: 2.09170840, Expectation = 1;
geary.test(ACM.area, ACM.nb2listw.rook, randomisation = FALSE)  # ACM geary r
ook: 2.134989922, Expectation = 1;

##### PART B #####
# Import dataset Crime_PA2002 shapefile
Crime_PA2002 <- readOGR(".", "Crime_PA2002")

# Construct neighbours list from polygon list
Crime_PA2002.rook <- poly2nb(Crime_PA2002, queen = FALSE)  # rook

# Spatial Weights for neighbours lists
Crime_PA2002.rook.nb2listw <- nb2listw(Crime_PA2002.rook)

##### Global G statistics
globalG.test(Crime_PA2002$BURG01, Crime_PA2002.rook.nb2listw)  # Global G sta
tistic: 2.974714e-02 ; Expectation: 1.515152e-02; Variance: 1.600546e-05

# GWR
DistanceMatrix <- gw.dist(dp.locat = coordinates(Crime_PA2002))  # the euclid
ian distance matrix
DistanceMatrixMifflin <- DistanceMatrix[61, ]
DistanceMatrixMifflin <- DistanceMatrixMifflin[-61]
weight <- 1/DistanceMatrixMifflin
weight

# spgwr package gwr
bw <- gwr.sel(INDEX01 ~ POP_CRI01 + AG_CRI01 + Area, data = Crime_PA2002[Crim
e_PA2002$COUNTY != "Mifflin County", ])
gwr.model <- gwr(INDEX01 ~ POP_CRI01 + AG_CRI01 + Area, data = Crime_PA2002[C
rime_PA2002$COUNTY != "Mifflin County", ], bandwidth = bw, hatmatrix = TRUE,
se.fit = TRUE, weights = weight)
gwr.model
x <- gwr(INDEX01~POP_CRI01+AG_CRI01+Area, data = Crime_PA2002, bandwidth = bw
, predict = TRUE, se.fit = TRUE, fittedGWRobject = gwr.model)
x$SDF$pred[61] # the predict result

##### PART C #####
PA_County_Select <- readOGR(".", "PA_County_Select")
Ozone_Value <- read.delim("Ozone_Value.dat", header = FALSE, sep = "|")
Ozone_Sensor_Locs <- readOGR(".", "Ozone_Sensor_Locs")
PA_County_Select@data$COUNTY <- as.character(PA_County_Select@data$COUNTY)
names(Ozone_Value)[3] <- "id"

## Inverse Distance Weighting (IDW)
```

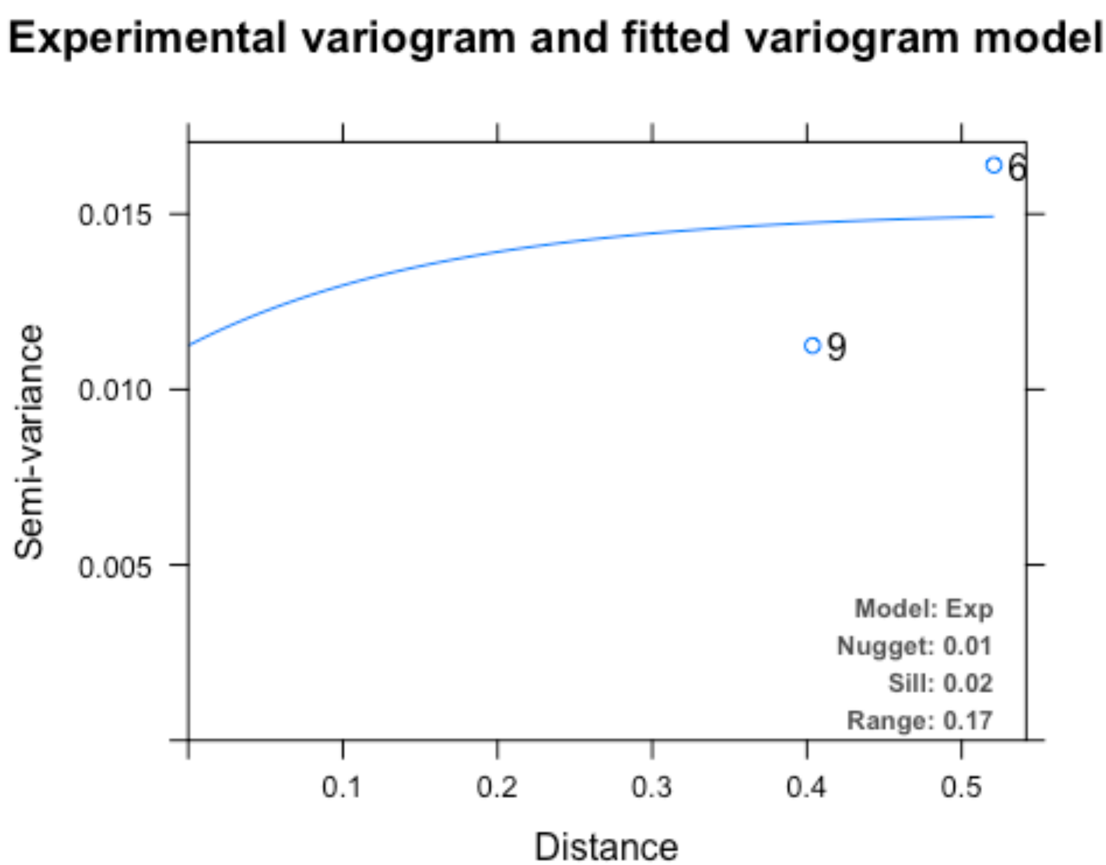
```r
centroids <- as.data.frame(coordinates(PA_County_Select))  # get the centroids for each county
names(centroids) <- c("centroidX", "centroidY")  # rename the column name
rownames(centroids) <- seq(1:length(centroids$centroidX))  # rename the row index
ControlPoints <- cbind.data.frame(Ozone_Sensor_Locs$id, Ozone_Sensor_Locs$long, Ozone_Sensor_Locs$lat)  # get all the points long and lat that would be the control points
names(ControlPoints) <- c("id", "controlX", "controlY")  # rename the column names
Sub_Ozone_Value <- Ozone_Value[Ozone_Value$id %in% ControlPoints$id, c(3, 6, 8)]  # get the ozone value with the same id in the control points
# Create the IDW calculation table in loop and record the results
Sub_Ozone_Value <- Sub_Ozone_Value[Sub_Ozone_Value$V6 == "OZONE", -2]  # get the ozone value
ControlPoints$id <- as.character(ControlPoints$id)  # change id to char
Sub_Ozone_Value$id <- as.character(Sub_Ozone_Value$id)  # change id to char
ControlPoints <- inner_join(ControlPoints, Sub_Ozone_Value)  # Join to get the ozone value for each sensor station
names(ControlPoints)[4] <- "ozone_value"
ResultTable <- as.data.frame(PA_County_Select@data$COUNTY)  # This Table will record the results
ResultTable$x <- centroids$centroidX  # long of centroid for each polygon
ResultTable$y <- centroids$centroidY  # lat of centroid for each polygon
ResultTable$z <- NA  # interpolated ozone value
# using for loop to interpolate for each polygon
for (i in 1:length(centroids$centroidX)) {
    centroidX <- ResultTable$x[i]
    centroidY <- ResultTable$y[i]
    IDWtable <- ControlPoints
    IDWtable$Distance <- sqrt((IDWtable$controlX - centroidX)^2 + (IDWtable$controlY - centroidY)^2)
    IDWtable <- IDWtable[order(IDWtable$Distance), ]
    rownames(IDWtable) <- seq(1:length(IDWtable$id))
    IDWtable <- IDWtable[1:5, ]  # keep only 5 nearest
    IDWtable$InverseDistance <- 1/IDWtable$Distance
    IDWtable$Weight <- IDWtable$InverseDistance/sum(IDWtable$InverseDistance)
    IDWtable$WeightedValue <- IDWtable$Weight * IDWtable$ozone_value
    ResultTable$z[i] <- sum(IDWtable$WeightedValue)
}
ResultTable <- ResultTable[, -1]
ResultTable

# Create the IDW Map
IDW.map <- as.data.frame(ResultTable)
coordinates(IDW.map) = ~x + y
xRange <- as.numeric(bbox(IDW.map)[1, ])
yRange <- as.numeric(bbox(IDW.map)[2, ])
grid <- expand.grid(x = seq(from = xRange[1], to = xRange[2], by = 0.01), y = seq(from = yRange[1], to = yRange[2], by = 0.01))
```

```
coordinates(grid) <- ~x + y
gridded(grid) <- TRUE
IDW.data <- gstat::idw(ResultTable$z ~ 1, locations = IDW.map, newdata = grid
)
OzonePlot <- par(mar = c(0, 0, 0, 0))
image(IDW.data, "var1.pred", col = terrain.colors(50))
contour(IDW.data, "var1.pred", add = TRUE, nlevels = 10
plot(IDW.map, add = TRUE, pch = 10, cex = 0.5)
text(coordinates(PA_County_Select), paste(as.character(round(ResultTable$z, 1
)), as.character(PA_County_Select$COUNTY)), pos = 4, cex = 0.8, col = "black"
)
map.axes(cex.axis = 0.8)
par(OzonePlot)

## Ordinary Kriging (OK)
Ozone_Value <- Ozone_Value[Ozone_Value$V6 == "OZONE", ]  # get all the ozone
value
Ozone_Sensor_Locs_data <- Ozone_Sensor_Locs@data
Ozone_Sensor_Locs_data$id <- as.character(Ozone_Sensor_Locs_data$id)
Ozone_Value$id <- as.character(Ozone_Value$id)
Ozone_Sensor_Value <- inner_join(x = Ozone_Sensor_Locs_data, y = Ozone_Value)
Sensor.num <- nrow(Ozone_Sensor_Value)  # there are 11 sensors has value
Ozone_Sensor_Value <- subset.data.frame(Ozone_Sensor_Value, select = c(long,
lat, V8))
names(Ozone_Sensor_Value) <- c("x", "y", "z")
centroids$z <- NA  # create the new column z to store the ozone value
names(centroids) <- c("x", "y", "z")
dataset <- rbind.data.frame(Ozone_Sensor_Value, centroids)

# Create matrix D of distance between control points
D <- as.data.frame(matrix(data = NA, nrow = Sensor.num, ncol = Sensor.num))
names(D) <- c(1:nrow(D))
for (i in 1:nrow(D)) {
    for (j in 1:nrow(D)) {
        D[i, j] = sqrt((Ozone_Sensor_Value$x[i] - Ozone_Sensor_Value$x[j])^2+
                   (Ozone_Sensor_Value$y[i] - Ozone_Sensor_Value$y[j])^2)
    }
}

# exponential semivariogram model
variogram = Ozone_Sensor_Value
coordinates(variogram) = ~x + y
variogram <- autofitVariogram(z ~ x + y, variogram, model = "Exp")  # exponen
tial
plot(variogram)  # sill is 15, range is 0.11, nugget is 14
```




```
a <- 0.11  # range
c <- 15  # sill
n <- 14
gamma <- function(d) {
    g <- n + (c - n) * (1 - exp(-d/a))
    return(g)
}

# get Matrix A
A <- gamma(D)
A <- rbind.data.frame(A, 1)
A <- cbind.data.frame(A, 1)
diag(A) <- 0
names(A) <- c(seq(1:12))
A.inverse <- solve(A)

for (i in 1:10) {
    # for each centroid get vector d, d
    d <- c()
    for (j in 1:11) {
        # for each control point
        distance <- sqrt((centroids$x[i] - Ozone_Sensor_Value$x[j])^2 + (cent
roids$y[i] - Ozone_Sensor_Value$y[j])^2)
```

```r
        d <- c(d, distance)
    }
    # get vector b
    b <- c(gamma(d), 1)
    # get the weight w
    w = A.inverse %*% b
    z.estimate <- sum(ControlPoints$ozone_value * w[1:11])
    centroids$z[i] = z.estimate
}
centroids

# create the OK map // parameters
OK.result <- as.data.frame(centroids)
coordinates(OK.result) = ~x + y
variogram <- autofitVariogram(z ~ x + y, OK.result, model = "Exp")
variogram
plot(variogram)
```




```r
nugget = 0.01
sill = 0.02
range = 0.17
model <- vgm(psill = sill, model = "Exp", range = range, nugget = nugget)
krige <- krige(OK.result$z ~ 1, OK.result, grid, model = model)  # using ordi
```

```
nary kriging
OzonePlot2 <- par(mar = c(0, 0, 0, 0))
image(krige, "var1.pred", col = terrain.colors(50))
contour(krige, "var1.pred", add = TRUE, nlevels = 10)
plot(OK.result, add = TRUE, pch = 10, cex = 0.5)
text(coordinates(OK.result), paste(as.character(round(OK.result$z, 1)), as.ch
aracter(PA_County_Select$COUNTY)), pos = 4, cex = 0.8, col = "black")
map.axes(cex.axis = 0.8)
par(OzonePlot2)
```