Spatial HW 4

David Coomes

March 2, 2020

Question 1

(a) Using Moran's statistic and standardized weights ("W" option) we do not find significant clustering of cancer in our data (p=0.177). The significance is reduced further if we adjust for latitude and longitude to control for large-scale trends (p=0.377). We see similar results (non-significance) using non-standardized weights ("B" option).

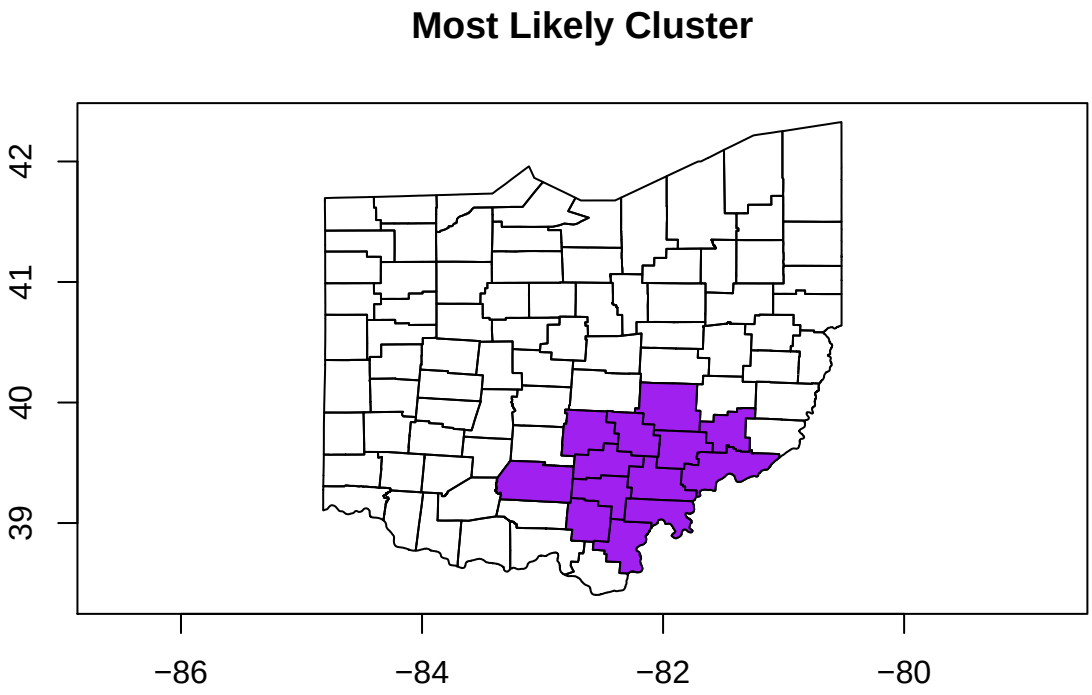
(b) Using Geary's test and standardized weights ("W" option), we find no evidence of cancer clustering using unadjusted residuals (p=0.208) or residuals adjusted for latitude and longitude (p=0.381). We see similar results (non-significance) when using non-standardized weights ("B" option).

Based on these two tests, there is no evidence of cancer clustering in these data.

Question 2

(a) Using the SatScan method of Kulldorff with a maximum population size of 20%, these data are significantly clustered (p=0.044).

Figure 1: Most likely cluster of cancer in Ohio using SatScan method (p=0.044).



Question 3

(a)

Figure 2: Cloud semi-variogram comparing variance of calcium content and distance between points using ca20 dataset.

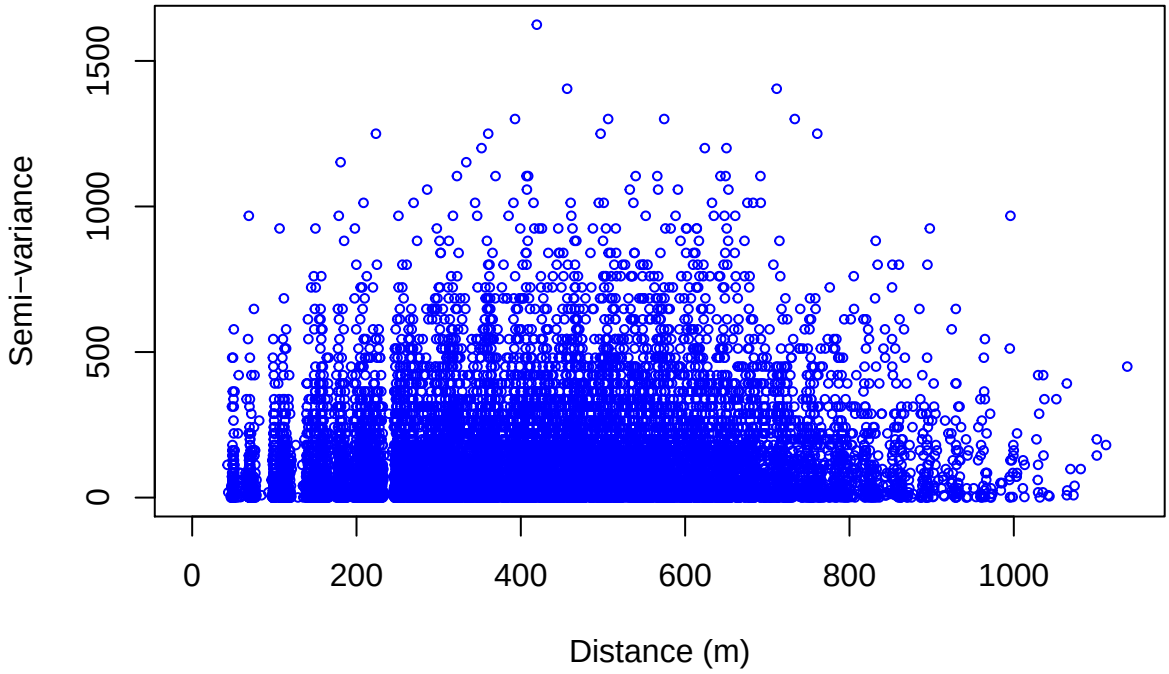
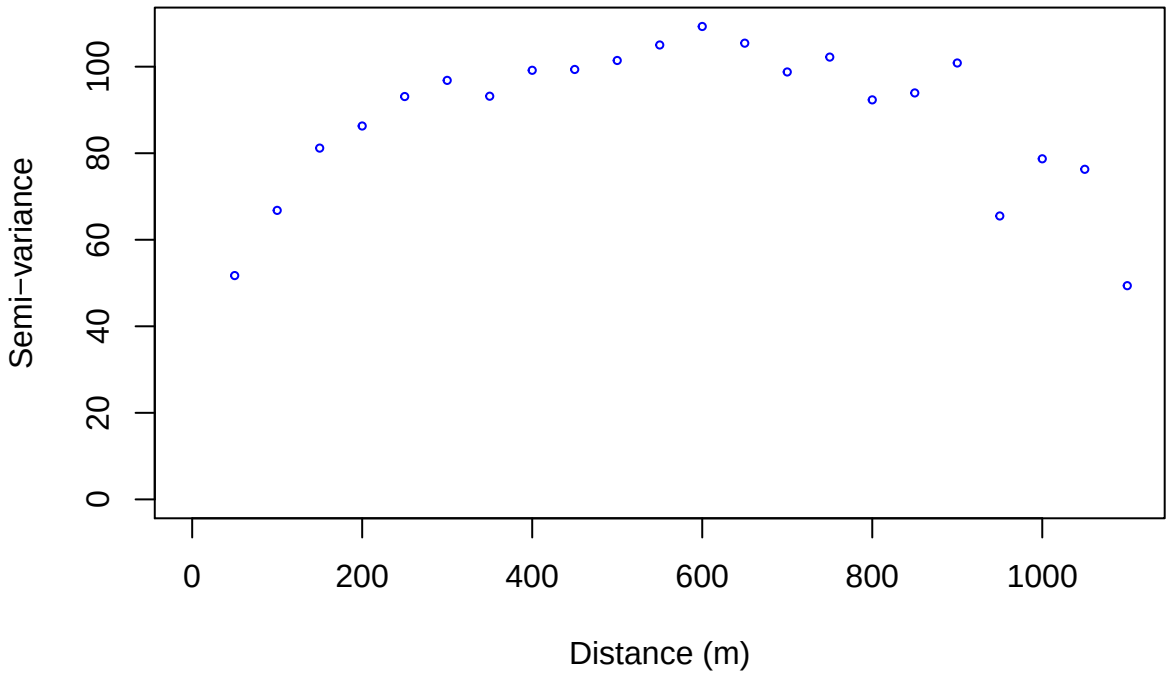


Figure 3: Binned semi-variogram comparing variance of soil calcium content and distance between points, adjusting for altitude and section.



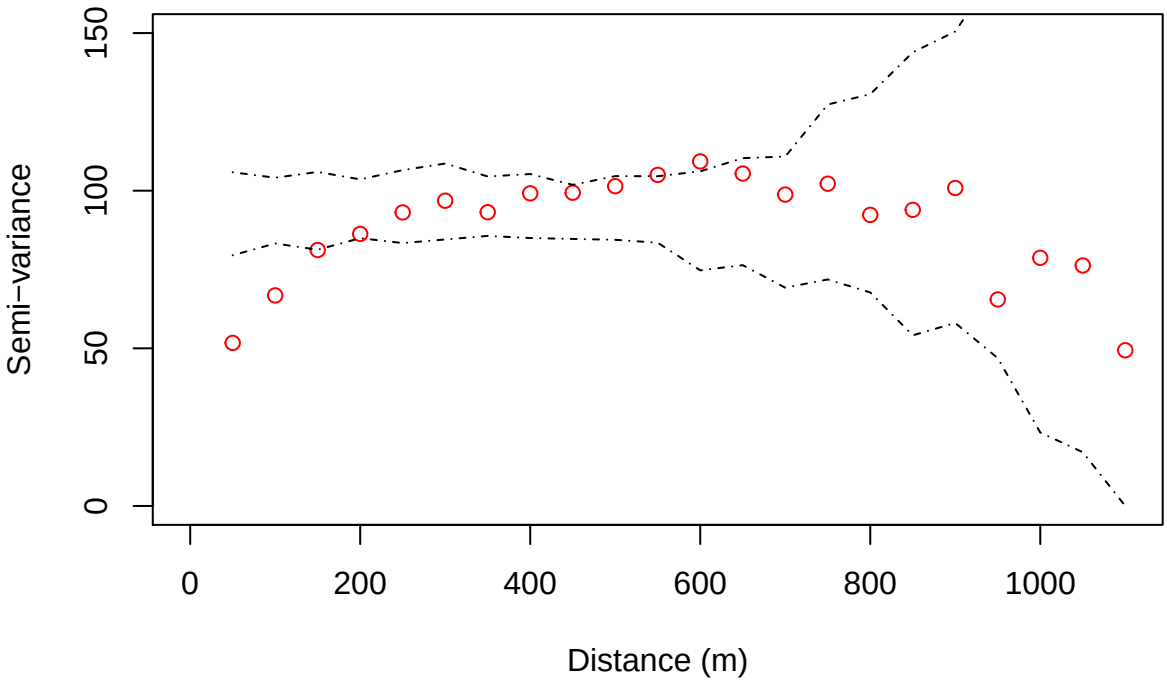
We cannot tell much about spatial dependence by looking at the cloud variogram - there is too much noise when we examine all points together. We can infer some information by examining the binned semi-variogram

(Figure 3). From this, we see some spatial dependence for points that are close together, and this spatial dependence weakens as points get further apart. We can see that some spatial dependence exists until the points are approximately 300 meters apart, at which point the curve flattens out.

(b)

Figure 4: Monte Carlo test of no spatial correlation (99 simulations).

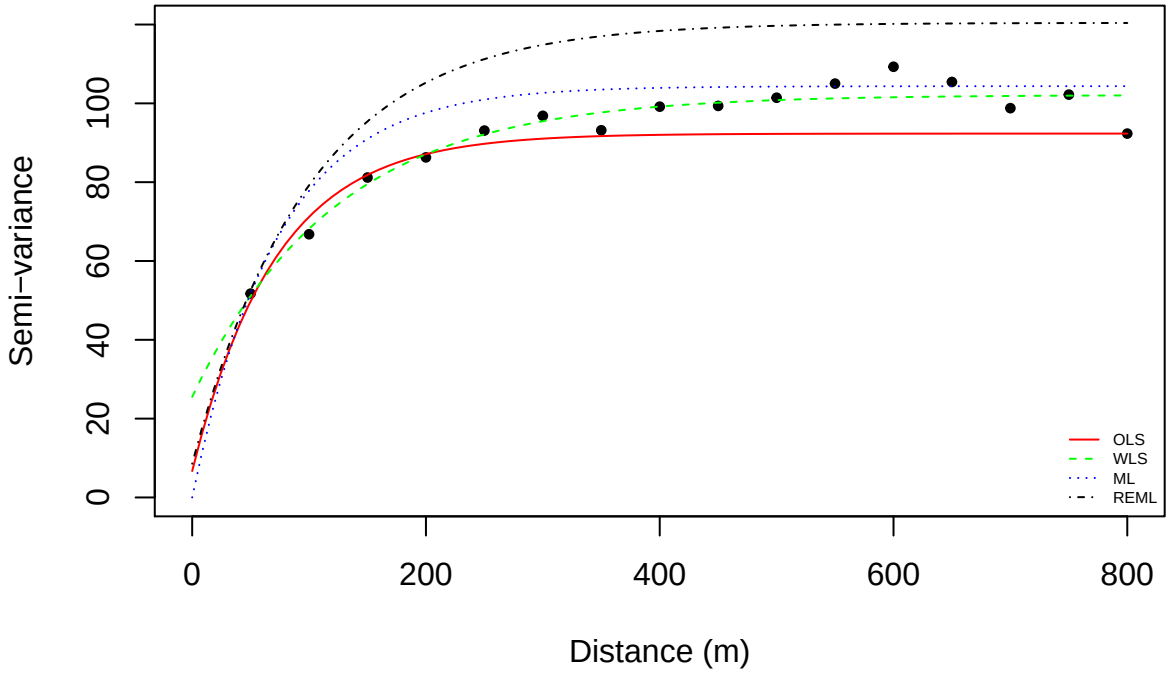
```
## variog.env: generating 99 simulations by permutating data values
## variog.env: computing the empirical variogram for the 99 simulations
## variog.env: computing the envelops
```



From the Monte Carlo simulation of no spatial correlation, we can see that there is likely some spatial correlation as the low semi-variance seen in points that are close together would be unlikely given random chance.

(c)

Figure 5: Exponential models fit to binned variogram using least squares and maximum likelihood estimation.



(d)

Figure 6: Prediction of calcium concentration using Kriging.

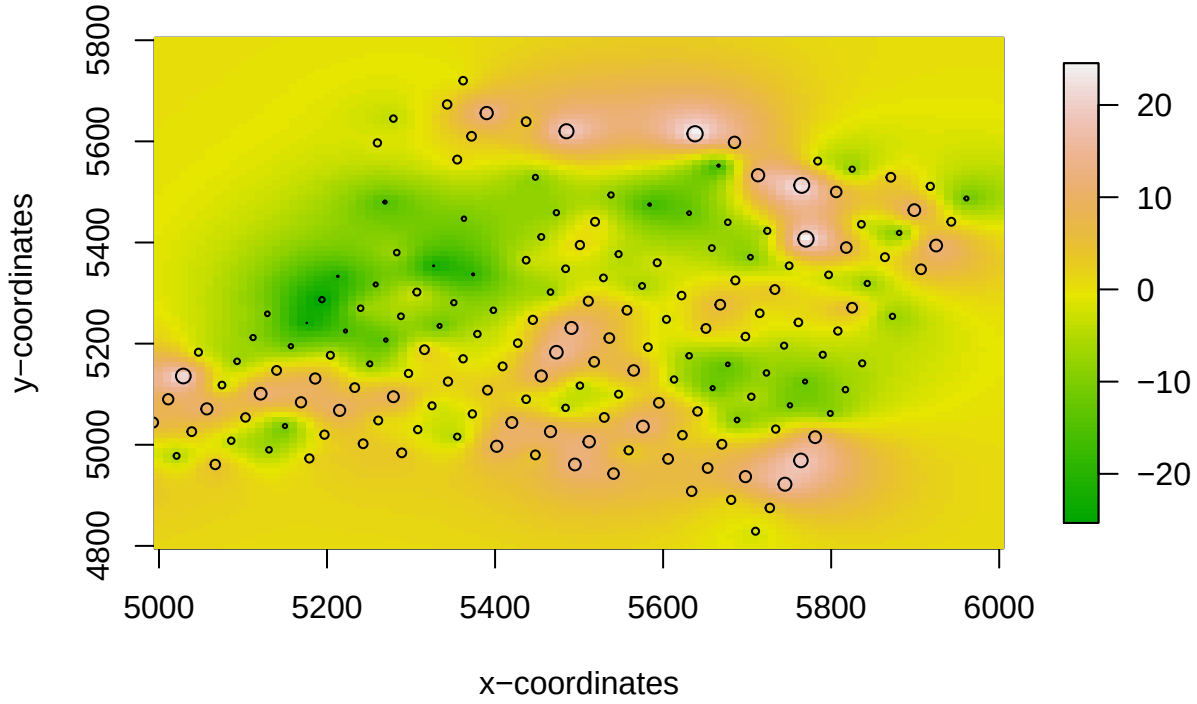
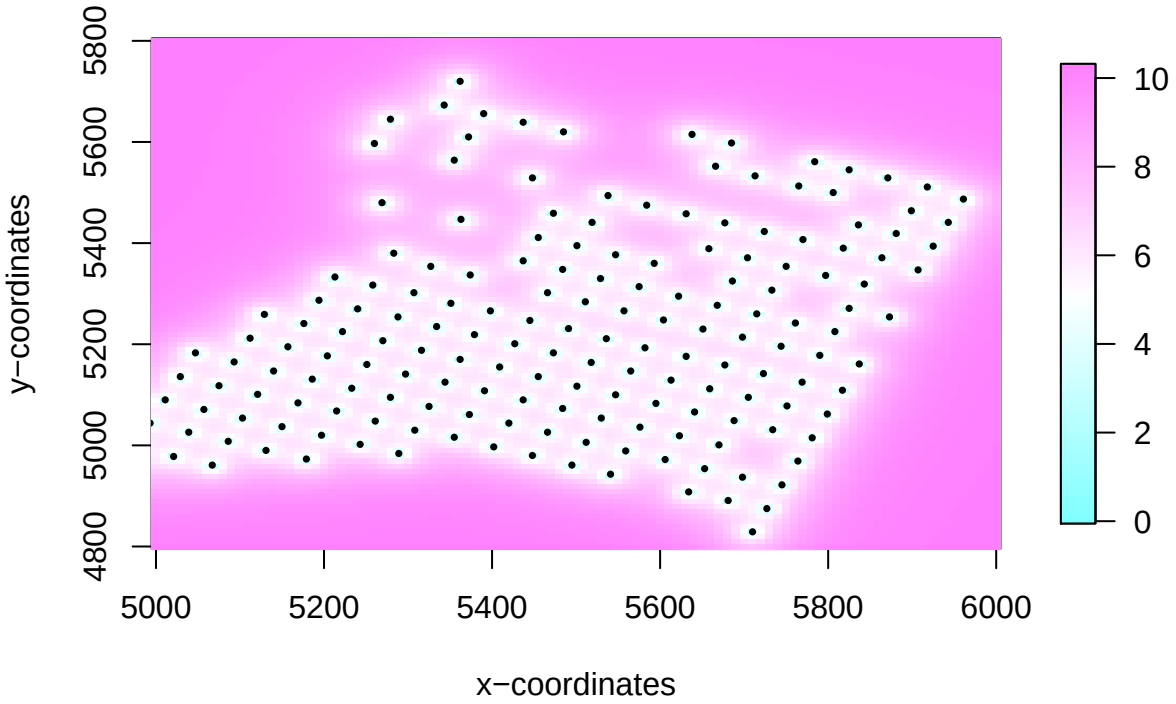


Figure 7: Standard deviation of estimates from prediction of calcium concentration using Kriging.



The prediction of calcium concentration using Kriging shows several high concentration patches that coincide with areas where multiple high-concentration measurements were found. The map of standard deviation shows low SEs around the measurements and more uncertainty in between the measurements, as we would expect.

Appendix

```
if (!require(knitr)) install.packages("knitr")

knitr::opts_chunk$set(echo = FALSE, warning=FALSE, cache=TRUE, message=FALSE)

if (!require(spdep)) install.packages("spdep", repos = "http://cran.us.r-project.org")
if (!require(raster)) install.packages("raster", repos = "http://cran.us.r-project.org")
if (!require(leaflet)) install.packages("leaflet", repos = "http://cran.us.r-project.org")
if (!require(SUMMER)) install.packages("SUMMER", repos = "http://cran.us.r-project.org")




if (!isTRUE(requireNamespace("INLA", quietly = TRUE))) {
  install.packages("INLA", repos = c(getOption("repos"),
           INLA = "https://inla.r-inla-download.org/R/stable"),
           dep=TRUE)
}


if (!require(SUMMER)) install.packages("SUMMER", repos = "http://cran.us.r-project.org")
if (!require(foreign)) install.packages("foreign", repos = "http://cran.us.r-project.org")
if (!require(haven)) install.packages("haven")
if (!require(rgeos)) install.packages("rgeos")
if (!require(rgdal)) install.packages("rgdal")
if (!require(maptools)) install.packages("maptools")
if (!require(sp)) install.packages("sp")
if (!require(spdep)) install.packages("spdep")
if (!require(SpatialEpi)) install.packages("SpatialEpi")
if (!require(RColorBrewer)) install.packages("RColorBrewer")
if (!require(ggplot2)) install.packages("ggplot2")
if (!require(maps)) install.packages("maps")
if (!require(broom)) install.packages("broom")
if (!require(raster)) install.packages("raster")
if (!require(leaflet)) install.packages("leaflet")
if (!require(dplyr)) install.packages("dplyr")
if (!require(tidyverse)) install.packages("tidyverse")
if (!require(SUMMER)) install.packages("SUMMER")



rm(list=ls())
library(haven)
library(INLA)
library(rgdal)
library(maptools)
library(sp)
library(spdep)
library(SpatialEpi)
library(RColorBrewer)
library(ggplot2)
library(maps)
```

```r
library(broom)
library(raster)
library(leaflet)
library(dplyr)
library(SUMMER)


knitr::opts_chunk$set(echo = FALSE)


rm(list=ls())
library(foreign)
library(haven)
library(INLA)
library(rgdal)
library(maptools)
library(sp)
library(spdep)
library(SpatialEpi)
library(RColorBrewer)
library(ggplot2)
library(maps)
library(broom)
library(raster)
library(leaflet)
library(dplyr)
library(tidyverse)
library(SUMMER)
library(rgeos)


#Cancer data

#using github
link = "https://github.com/dmccoomes/Spatial_epi/raw/master/HW%203/Data/ohio_2019_version.txt"
ohio_canc <- read.table(url(link), header=TRUE)

#pc in office
#ohmap <- readOGR(dsn="C:\\Users\\dcoomes\\Dropbox\\Classes\\Spatial modeling\\HW 3\\Data\\Map data", l

#laptop
#ohmap <- readOGR(dsn="/Users/david/Documents/GitHub/Spatial_epi/HW 3/Map data", layer="ohio_map")

#computer in library
#ohmap <- readOGR(dsn="C:\\Users\\dcoomes\\Documents\\GitHub\\Spatial_epi\\HW 3\\Map data", layer="ohio

#computer at home
setwd("/Users/david/Documents/GitHub/Spatial_epi")
ohmap <- readOGR(dsn="HW 3/Map data", layer="ohio_map")

#ordering of regions is not the same among the data sets - how do we align these two?
#summary(ohmap)
#ohmap$COUNTYFP00
```

```r
#creating neighbor file
nb.map <- poly2nb(ohmap)



set.seed(03022020)



col.W <- nb2listw(nb.map, style="W",zero.policy=TRUE)
col.B <- nb2listw(nb.map, style="B",zero.policy=TRUE)

#Merging map and cancer data together to include lat and lon in cancer data
ohio_canc2 <- ohio_canc
ohmap$fips <- as.numeric(as.character(ohmap$CNTYIDFP00))
ohmap$lat <- ohmap$INTPTLAT00
ohmap$lon <- ohmap$INTPTLON00
ohio_canc2 <- merge(ohio_canc2, ohmap[, c("fips", "lat", "lon")], by.x="fips", by.y="fips")
#ohio_canc2 <- merge(ohio_canc2, ohmap, by="fips")
#ohio_canc2$lat <- as.character(as.numeric(ohio_canc2$lat))
#ohio_canc2$lon <- as.character(as.numeric(ohio_canc2$lon))

ohio_canc2$lat <- as.numeric(as.character(ohio_canc2$lat))
ohio_canc2$lon <- as.numeric(as.character(ohio_canc2$lon))


# kappaval <- function(Obs, fitted, df) {
#   sum((Obs - fitted)^2/fitted)/df
# }

#fitting poisson model without covariates
quasip_mod <- glm(data=ohio_canc, Obs ~ 1, offset=log(Exp), family=quasipoisson())
sidsres <- residuals(quasip_mod, type="pearson")
moran.test(sidsres, col.W)

#running Moran test using B weight option
moran.test(sidsres, col.B)


#fitting poisson model with covariates
quasip_mod.1 <- glm(data=ohio_canc2, Obs ~ lat + lon, offset=log(Exp), family=quasipoisson())
summary(quasip_mod.1)         #latitude is higly significant (p=0.0018), lon is not (p=0.132)
sidsres.1 <- residuals(quasip_mod.1, type="pearson")
moran.test(sidsres.1, col.W)

moran.test(sidsres.1, col.B)


# kappaest <- kappaval(ohio_canc$Obs, mod$fitted, mod$df.resid)
# nMC <- 1000
# ncts <- length(ohio_canc$Exp)
# yMC <- matrix(rpois(n=nMC * ncts, lambda=ohio_canc$Exp),
```

```
#               nrow=ncts, ncol=nMC)
# kappaMC <- NULL
# for (i in 1:nMC){
#  modMC <- glm(yMC[,i]~1,offset=log(ohio_canc$Exp),family="quasipoisson")
#  kappaMC[i] <- kappaval(yMC[,i],modMC$fitted,modMC$df.resid)
# }


#Geary test on non-adjusted data
geary.test(sidsres,col.W)
geary.test(sidsres, col.B)

#Geary test on adjusted data
geary.test(sidsres.1, col.W)
geary.test(sidsres.1, col.B)



library(SpatialEpi)
pop.upper.bound <- 0.2
n.simulations <- 5000
alpha.level <- 0.05

#need to create this - slide 39?
#geo <- latlong2grid(coordinates(ohmap))

#This is a map of NC, not ohio - why is that?
ohmap2 <- readShapeSpatial(system.file("shapes/sids.shp",
                package = "maptools")[1], IDvar = "FIPSNO",
                proj4string = CRS("+proj=longlat +ellps=clrk66"))


#first,  form a matrix containing the centroids
# getLabelPoint <- function(county) {
# Polygon(county[c("long", "lat")])@labpt
# }


#other way to create centroids - is this the same way and which one is necessary for this exercise
centroids <- latlong2grid(coordinates(ohmap))

# #create dataframe from map data
# oh_df <- map_data("county", "ohio")
# oh_df <- map_data("county", "ohio")
# centOH <- by(oh_df, oh_df$subregion, getLabelPoint)
# centOH <- do.call("rbind.data.frame", centOH)
# names(centOH) <- c("long", "lat")
#
# centroids <- matrix(0, nrow=n, ncol=2)
# for (i in 1:n) {
# centroids[i, ] <- c(centOH$lat[i], centOH$long[i])
# }
```

```r
Kpoisson <- kulldorff(centroids, ohio_canc2$Obs, population=ohio_canc2$N,
                      expected.cases=NULL,
                      pop.upper.bound, n.simulations, alpha.level, plot=T)


Kcluster <- Kpoisson$most.likely.cluster$location.IDs.included




#Why is this not working?
plot(ohmap, axes=TRUE)
plot(ohmap[Kcluster,], add=TRUE, col="purple")
title("Most Likely Cluster")


if (!require(geoR)) install.packages("geoR")
library(geoR)
data("ca20")
plot(ca20)


data("meuse")
zmat <- matrix(cbind(meuse$x, meuse$y, log(meuse$zinc)),
               ncol=3, nrow=155, byrow=F)
geozinc <- as.geodata(zmat, coords.col = c(1, 2),
                      data.col=c(3))


cloudca <- variog(ca20, option="cloud")
plot(cloudca, ylab="Semi-variance", xlab="Distance (m)",
     col="blue", cex=0.6)


#this works, but do we need to include some trend variables? I'm not sure what those are in this datase
binca <- variog(ca20, uvec=seq(0, 1200, 50))
plot(binca, ylab="Semi-variance", xlab = "Distance (m)",
     cex=0.5, col="blue")


#controlling for altitude and exposure - this is probably more useful
binca.1 <- variog(ca20, uvec=seq(0, 1200, 50),
                  trend = ~ca20$covariate$altitude + ca20$covariate$area)
plot(binca.1, ylab="Semi-variance", xlab = "Distance (m)",
     cex=0.5, col="blue")


set.seed(03092020)
ca.env <- variog.mc.env(ca20, obj=binca.1)
plot(binca.1, env=ca.env, col="red", xlab="Distance (m)", ylab="Semi-variance", ylim =c(0, 150))


#calculating theta using estimate of d=175
```

```r
theta <- (-300)/(log(0.05))

#where do we get the numbers for the fit in this model - do they both come from the binned variogram
olsfit <- variofit(binca.1, ini=c(50, theta), weights="equal")
olsfit

#weighted least squares
wlsfit <- variofit(binca.1, ini=c(50, theta))
wlsfit

#maximum likelihood
mlfit <- likfit(ca20, ini=c(50, theta), trend= ~ca20$covariate$altitude + ca20$covariate$area)

#restricted maximum likelihood
remlfit <- likfit(ca20, ini=c(50, theta), lik.method="RML", trend= ~ca20$covariate$altitude + ca20$cova



plot(binca.1, max.dist = 800, xlab="Distance (m)",
     ylab="Semi-variance", pch=19, cex=0.6, ylim=c(0, 120))
lines(olsfit, max.dist=800, col="red")
lines(wlsfit, max.dist=800, lty=2, col="green")
lines(mlfit, max.dist=800, lty=3, col="blue")
lines(remlfit, max.dist=800, lty=4, col="black")
legend("bottomright", legend=c("OLS", "WLS", "ML", "REML"), lty=c(1,2,3,4), bty="n", col=c("red", "gree


points(ca20, pt.divide="data.proportional", cex.min=0.05,
       cex.max=0.4, xlab="x-coordinate", ylab="y-coordinate",
       col="green")


lmfit <- lm(ca20$data ~ ca20$covariate$altitude + ca20$covariate$area)
lmfit

detrend <- as.geodata(cbind(ca20$coords, lmfit$residuals))


points(detrend, pt.divide="rank.prop", xlab="x-coordinate", ylab="y-coordinate", cex.min=0.1, cex.max=0


#using the same parameters from the maximum likelihood above - not sure how to get these parameters
mlfit2 <- likfit(detrend, ini=c(50, theta))



pred.grid <- expand.grid(seq(5000, 6000, l=101), seq(4800, 5800, l=101))


kc <- krige.conv(detrend, loc=pred.grid,
                 krige = krige.control(obj.m=mlfit2))
```

```r
library(fields)
image.plot(x=pred.grid[["Var1"]][1:101], y=unique(pred.grid[["Var2"]]),
  z=matrix(kc$predict, nrow=101, ncol=101), col=terrain.colors(100),
  xlab= "x-coordinates", ylab="y-coordinates")
symbols(detrend$coords[, 1], detrend$coords[, 2], circles=(detrend$data-min(detrend$data))/1, add=T, in



image.plot(x=pred.grid[["Var1"]][1:101], y=unique(pred.grid[["Var2"]]),
           z=matrix(sqrt(kc$krige.var), nrow=101, ncol=101),
           col=cm.colors(100), xlab="x-coordinates", ylab="y-coordinates")
points(detrend$coords[,1], detrend$coords[,2], pch=16, cex=0.5)
```