MTH 324, Optional Lab #1, Fall 2023 Name _____

Instructions: Follow along with the tutorial portion of the lab. Replicate the code examples in R on your own, along with the demonstration. Then use those examples as a model to answer the questions/perform the tasks that follow. Copy and paste the results of your code to answer questions where directed. Submit your response file and the code used (both for the tutorial and part two). Your code file and your lab response file should each include your name inside.

Spatial Statistics

In this lab, we are going to look at computing some simple spatial statistics. Before loading any libraries used in these examples, be sure to install them first.

To begin, we are going to load some data from the spatstat package and plot one of the datasets.

```
library(spatstat)
data(swedishpines)
X <- swedishpines
plot(X)
axis(1)
axis(2)
summary(X)
```

This code will plot the data as points on an x-y grid. We can also plot a density function instead.

```
den <- density(x = X, sigma = 10)
summary(den)
plot(den, main = "Intensity")
contour(den, add = TRUE)
```

One method of analyzing these points is to determine if the data is spatially random, evenly distributed or clustered. To test this, we can conduct a quadrat analysis. To do this, we first divide the region into a grid.

```
Q <- quadratcount(X, nx = 4, ny = 3)
Q
```

Then we can replot our data with the grid and counts overlaying the plot.

```
plot(X)
axis(1)
axis(2)
plot(Q, add = TRUE, cex = 2)
```

A quadrat test on this grid is essentially a χ^2-test that assumes that each section of the grid is homogeneous with respect to the number of expected points. The default of the quadrat test is two-sided.

```
quadrat.test(Q)
```

Since our null hypothesis is that the points are essentially random, and p-value above 0.05 means that we don't have strong evidence against that claim.

We can also test to see if the data takes other form such as "regular" or "clustered". This changes our assumptions and thus, potentially, the results of the test.

```
quadrat.test(Q, alternative = "two.sided") #default
quadrat.test(Q, alternative = "regular")
quadrat.test(Q, alternative = "clustered")
```

Let's look at some maps and other types of spatial analysis. Let's load in a new package, sf, and some data.

```
library(sf)
nameshp <- system.file("shape/nc.shp", package = "sf")
d <- st_read(nameshp, quiet = TRUE)
d$vble <- d$SID74
d$vble2 <- d$SID79
```

Then we'll plot it with ggplot. We get a map of North Carolina counties with counts of sudden infant deaths from 1974.

```
library(ggplot2)
library(viridis)
ggplot(d) + geom_sf(aes(fill = vble)) +
  scale_fill_viridis() + theme_bw()
```

There are a number of other packages that can be used to make maps. These are described in reference [2] below.

We can look at nearest neighbor relationships. We'll import some data to look at neighborhoods in Columbus, Ohio.

```
library(spData)
library(spdep)
map <- st_read(system.file("shapes/columbus.shp",
                           package = "spData"), quiet = TRUE)
```

We can get a glimpse of the data (as numbers) and plot the data with the code below.

```
nb <- spdep::poly2nb(map, queen = TRUE)
head(nb)
plot(st_geometry(map), border = "lightgray")
plot.nb(nb, st_geometry(map), add = TRUE)
```

We can visualize the nearest neighbors by choosing one neighborhood and color-coding the surrounding neighborhoods by closeness.

```
id <- 20 # area id
map$neighbors <- "other"
map$neighbors[id] <- "area"
map$neighbors[nb[[id]]] <- "neighbors"
ggplot(map) + geom_sf(aes(fill = neighbors)) + theme_bw() +
  scale_fill_manual(values = c("gray30", "gray", "white"))
```

Experiment with the code by changing the id number and seeing how the map changes.

We can calculate neighborhoods by a number of different factors. For instance, we can base it on distance, or on other neighbors, or other measurements. Let's consider the neighbors approach first, then distance (from the center of the neighborhood).

```
coo <- st_centroid(map)
nb <- knn2nb(knearneigh(coo, k = 3)) # k number nearest neighbors
plot(st_geometry(map), border = "lightgray")
plot.nb(nb, st_geometry(map), add = TRUE)

nb <- dnearneigh(x = st_centroid(map), d1 = 0, d2 = 0.4)
plot(st_geometry(map), border = "lightgray")
plot.nb(nb, st_geometry(map), add = TRUE)
```

Compare the two maps produced. How do they differ?

Lastly, we'll look at some geostatistics. We'll need to load another package and then look at the data.

```
library(geoR)
d <- st_as_sf(data.frame(x = parana$coords[, 1],
                         y = parana$coords[, 2],
                         value = parana$data),
              coords = c("x", "y"))

ggplot(d) + geom_sf(aes(color = value), size = 2) +
  scale_color_gradient(low = "blue", high = "orange") +
  geom_path(data = data.frame(parana$border), aes(east, north)) +
  theme_bw()
```

The map displays measured rainfall amounts in a region of Brazil. One common geostatistical plot is the variogram.

```
plot(variog(coords = st_coordinates(d), data = d$value,
            option = "cloud", max.dist = 400))

plot(variog(parana, max.dist = 400))
```

Let's look at an example where we can use kriging to fill in the data for the regions that were not directly measured. We'll use a different data set and map package for this example.

```
library(sp)
library(gstat)
library(mapview)

data(meuse)
data(meuse.grid)

meuse <- st_as_sf(meuse, coords = c("x", "y"), crs = 28992)
mapview(meuse, zcol = "zinc",  map.types = "CartoDB.Voyager")

meuse.grid <- st_as_sf(meuse.grid, coords = c("x", "y"),
                       crs = 28992)
mapview(meuse.grid,  map.types = "CartoDB.Voyager")
```

Then we create the variogram and model the data. Here we'll do it with a simulation and then plot the results.

```
v <- variogram(log(zinc)~1, meuse)
m <- fit.variogram(v, vgm(1, "Sph", 300, 1))
plot(v, model = m)

sim <- krige(formula = log(zinc)~1, meuse, meuse.grid, model = m,
             nmax = 10, beta = 5.9, nsim = 5)

plot(sim)
```

Tasks:

1. For the swedishpines dataset, redo the quadrat analysis with different grid sizes a) 2x3, b) 5x5. Does this change your analysis at all?

2. Use the japanesepines dataset and repeat the same analysis we did with the swedishpines dataset. Include the graphs and describe the results of the quadrat test (you don't need to try alternate grid sizes).

3. For the North Carolina data, replot the map with the 1979 values instead of the 1974 values. Paste the map below.

4. For the Columbus neighborhood data, replot the nearest neighborhoods to a selected neighborhood, but changing the id <- 20 selection to 5 other numbers (between 0 and 49). Update the color scheme on each resulting plot: for example, use blue or green instead of grey, etc. Paste the maps below.

5. Recreate the two types of centroid maps for Columbus by changing a) the number of nearest neighbors (try one smaller number and one larger number), and b) the d1 and d2 distances. How do these changes affect the maps. Paste the maps below.

6. Starting with the parana data, try to recreate the interpolation model we did with the meuse data. (You may need to alter the crs information in the example.)

7. Use the View() function to look at each of the datasets we used in the models (swedishpines, parana, nc and columbus—some names were changed after we opened them), and describe how the spatial information is represented in the dataset in each case. Is there a column for x and y? Does the data have a geometry column? Is the geometry column a point or a polygon? Are the regions in the dataset named or are they identified in another way? (You should have a paragraph or 3 of description and explanation here.)

Resources:

1. https://www.paulamoraga.com/book-spatial/spatial-point-patterns.html
2. https://www.paulamoraga.com/book-spatial/making-maps-with-r.html
3. https://www.paulamoraga.com/book-spatial/geostatistical-data-1.html
4. https://www.paulamoraga.com/book-spatial/kriging.html
5. https://r-spatial.github.io/gstat/reference/predict.gstat.html
6. https://gsp.humboldt.edu/olm/R/04_01_Variograms.html
7. https://r.geocompx.org/adv-map.html
8. https://towardsdatascience.com/geostatistics-in-practice-using-r-4b7d32b7840d
9. https://rpubs.com/nabilabd/118172
10. https://search.r-project.org/CRAN/refmans/spatstat.geom/html/nndist.html