Problem Set 3

Shisham Adhikari

Math 345

Due: Before class on Thursday, 02/18/2021

Collaborators: Grayson White, Josh, DataCamp

Goals of this lab

1. Start exploring some real world spatial data
2. Spatial Autocorrelation, several ways

Problem 1

Get some data from the Census API!

a) We'll be using the tidycensus package today. Load it, and any other packages you need here:

```
pacman::p_load(tidycensus, tidyverse, spdep)
```

b) To use tidycensus, you'll need to get an API key. There are instructions here, along with some helpful details for how to use the package.

```
census_api_key("e20f7e545ee9474d9d353d401cc0e1d00d31deeb")
```

c) Download some census data using the "get_acs" function. You're going to make a few choices here:

- Choose an attribute that you want to analyze from the ACS. The link from part (b) also has information about how to get a list of available variables using the "load_variables" function.
- Pick the State that you'd like to analyze.
- Pick your spatial unit (your "geography" input for the "get_acs" function). Pick a unit such that you have at least 20 observations.

```
or <- get_acs(geography = "county",
              variables = c(rentburden = "B25071_001"),
              state = "OR",
              year = 2019,
              geometry = TRUE)
```

```
##   |                                                                               |
```

```
head(or)
```

```
## Simple feature collection with 6 features and 5 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -124.5662 ymin: 41.99562 xmax: -117.266 ymax: 45.86107
## CRS:            4269
##   GEOID                    NAME   variable estimate moe
## 1 41017 Deschutes County, Oregon rentburden     30.4 1.7
## 2 41003    Benton County, Oregon rentburden     36.1 2.4
## 3 41015     Curry County, Oregon rentburden     30.3 3.4
## 4 41061     Union County, Oregon rentburden     26.3 2.3
## 5 41055   Sherman County, Oregon rentburden     28.1 5.7
## 6 41051 Multnomah County, Oregon rentburden     30.7 0.4
##                         geometry
## 1 MULTIPOLYGON (((-122.0019 4...
## 2 MULTIPOLYGON (((-123.8167 4...
## 3 MULTIPOLYGON (((-124.3239 4...
## 4 MULTIPOLYGON (((-118.6978 4...
## 5 MULTIPOLYGON (((-121.0312 4...
## 6 MULTIPOLYGON (((-122.9292 4...
```

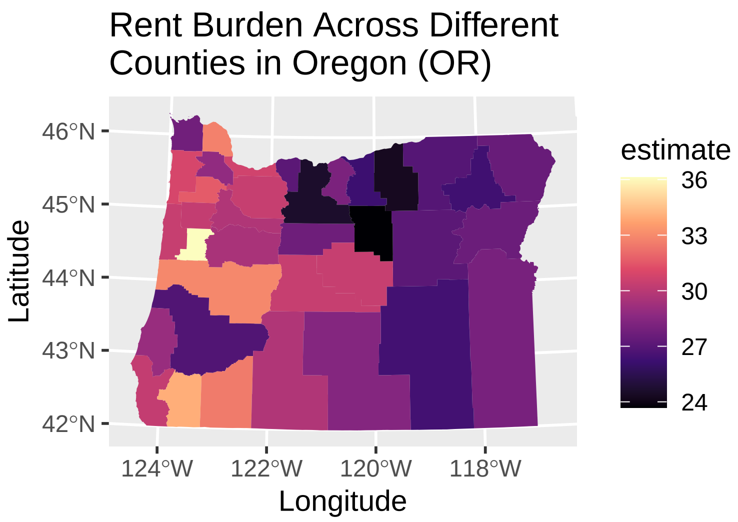
Problem 2

Vizualize your data! There is information on how to do this in ggplot here.

```r
library(crsuggest)
options(tigris_use_cache = TRUE)
#using crssuggest to identify an appropriate projected coordinate system for OR
suggest_top_crs(or)
```

```
## [1] 2991
```

```r
or %>%
  ggplot(aes(fill = estimate)) +
  geom_sf(color = NA) +
  coord_sf(crs = 2991) +
  scale_fill_viridis_c(option = "magma")+
  labs(x="Longitude", y="Latitude", title="Rent Burden Across Different \nCounties in Oregon (OR)")
```



Does your data appear to exhibit positive / negative spatial autocorrelation? Is it as you expected?

- The data appears to exhibit positive spatial autocorrelation for the most parts. We see that almost all counties in eastern Oregon have lower rent burden whereas most counties in western Oregon have higher rent burden. This is what I expected because rent burden determined by rent cost, income, and rentals are all likely to be spatially autocorrelated. It would also make sense for bigger cities like Portland and cities nearby to have higher rent burden than the rural parts of Oregon. Also, proximity to urban parts is highly likely to make the rent burden spatially autocorrelated.

Problem 3

Now let's calculate spatial autocorrelation statistics.

a) Create a rook's adjacency matrix.

```
#Create a Rooks' case neighborhood object
wr <- poly2nb(or, row.names=or$ID, queen=FALSE)
wm <- nb2mat(wr, style='B')
# Inspect the first order rook's adjacency matrix
wm[1:10, 1:10]
```

```
##    [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## 1     0    0    0    0    0    0    0    0    1     1
## 2     0    0    0    0    0    0    0    0    0     1
## 3     0    0    0    0    0    0    0    1    0     0
## 4     0    0    0    0    0    0    0    0    0     0
## 5     0    0    0    0    0    0    0    0    0     0
## 6     0    0    0    0    0    0    0    0    0     0
## 7     0    0    0    0    0    0    0    0    0     0
## 8     0    0    1    0    0    0    0    0    0     0
## 9     1    0    0    0    0    0    0    0    0     0
## 10    1    1    0    0    0    0    0    0    0     0
```

b) Calculate Moran's I and do a monte carlo simulation to test for signficance.

```
ww <- nb2listw(wr, style='B')
i1 <- moran(or$estimate, ww, n=length(ww$neighbours), S0=Szero(ww))$I
i1
```

```
## [1] 0.4055471
```

```
#Now we can test for significance
moran.mc(or$estimate, ww, nsim=111)
```

```
##
##  Monte-Carlo simulation of Moran I
##
## data:  or$estimate
## weights: ww
## number of simulations + 1: 112
##
## statistic = 0.40555, observed rank = 111, p-value = 0.008929
## alternative hypothesis: greater
```

c) Find Moran's I using the 2nd-5th order adjacency matrices.

- Here, the main idea behind constructing the 2nd-5th order adjacency matrices is to multiply the first order matrix with itself as many time as the order and replace all the positive values with 1 as we are working with unweighted networks only. Also, we need to make all the diagonal elements are 0 since we donot want seld-edges while looking into autocorrelation.

```
# Compute the second order matrix
SecondOrderMatrix_adj <- wm %*% wm
# Adjust the second order matrix
SecondOrderMatrix <- ((SecondOrderMatrix_adj) > 0) + 0
diag(SecondOrderMatrix) <- 0
m2 <- mat2listw(SecondOrderMatrix)
i2 <- moran(or$estimate, m2, n=length(m2$neighbours), S0=Szero(m2))$I
i2
```

```
## [1] 0.3008241
```

```
#Now repeat the same for higher orders
ThirdOrderMatrix_adj <- wm %*% wm %*% wm
ThirdOrderMatrix <- ((ThirdOrderMatrix_adj) > 0) + 0
diag(ThirdOrderMatrix) <- 0
m3 <- mat2listw(ThirdOrderMatrix)
i3 <- moran(or$estimate, m3, n=length(m3$neighbours), S0=Szero(m3))$I
i3
```

```
## [1] 0.1596037
```

```
FourthOrderMatrix_adj <- wm %*% wm %*% wm %*% wm
FourthOrderMatrix <- ((FourthOrderMatrix_adj) > 0) + 0
diag(FourthOrderMatrix) <- 0
m4 <- mat2listw(FourthOrderMatrix)
i4 <- moran(or$estimate, m4, n=length(m4$neighbours), S0=Szero(m4))$I
i4
```

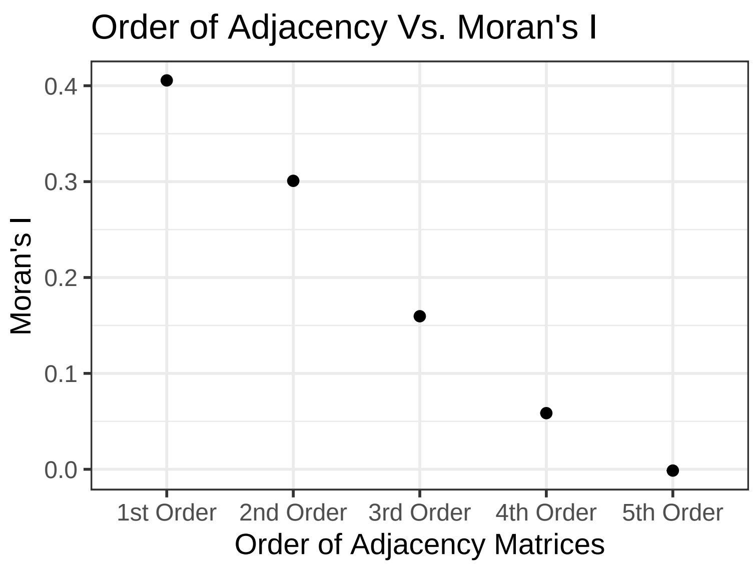
```
## [1] 0.05853298
```

```
FifthOrderMatrix_adj <- wm %*% wm %*% wm %*% wm %*% wm
FifthOrderMatrix <- ((FifthOrderMatrix_adj) > 0) + 0
diag(FifthOrderMatrix) <- 0
m5 <- mat2listw(FifthOrderMatrix)
i5 <- moran(or$estimate, m5, n=length(m5$neighbours), S0=Szero(m5))$I
i5
```

```
## [1] -0.001359755
```

d) Graph the Moran's I value you found for the 1st-5th order adjacency matrices. Is the result as you expect? Why?

```
moran.data <- data.frame(x = c("1st Order","2nd Order","3rd Order","4th Order","5th Order"),
                         y = c(i1, i2, i3, i4, i5))
ggplot(moran.data, aes(x,y))+
  geom_point() +
  theme_bw() +
  labs(x="Order of Adjacency Matrices",
       y="Moran's I", title =
         "Order of Adjacency Vs. Moran's I")
```



The result is what I expect because we observe that higher order means lower values of Moral I, meaning as you go further away from a point, you get less and less autocorrelation and this makes sense intuitively.

e) Calculate Geary's C using an inverse distance weighting matrix. Do you reach a similar conclusion as you do from the analysis in parts (a) - (d)? Why or why not?

```
#For first order adjacency
geary.test(or$estimate, listw = ww)
```

```
##
##  Geary C test under randomisation
##
## data:  or$estimate
## weights: ww
##
## Geary C statistic standard deviate = 3.172, p-value = 0.000757
## alternative hypothesis: Expectation greater than statistic
## sample estimates:
## Geary C statistic       Expectation          Variance
##        0.58885224        1.00000000        0.01680104
```

```
#For second order adjacency
geary.test(or$estimate, listw = m2)
```

```
##
##  Geary C test under randomisation
##
## data:  or$estimate
## weights: m2
##
## Geary C statistic standard deviate = 3.1905, p-value = 0.0007102
## alternative hypothesis: Expectation greater than statistic
## sample estimates:
## Geary C statistic       Expectation          Variance
##       0.690024243       1.000000000       0.009439319
```

```
#For third order adjacency
geary.test(or$estimate, listw = m3)
```

```
##
##  Geary C test under randomisation
##
## data:  or$estimate
## weights: m3
##
## Geary C statistic standard deviate = 2.0015, p-value = 0.02267
## alternative hypothesis: Expectation greater than statistic
## sample estimates:
## Geary C statistic       Expectation          Variance
##       0.841268759       1.000000000       0.006289186
```

```
#For fourth order adjacency
geary.test(or$estimate, listw = m4)
```

```
##
##  Geary C test under randomisation
##
## data:  or$estimate
## weights: m4
##
## Geary C statistic standard deviate = 1.2752, p-value = 0.1011
## alternative hypothesis: Expectation greater than statistic
## sample estimates:
## Geary C statistic       Expectation          Variance
##       0.926546140       1.000000000       0.003317884
```

```
#For fifth order adjacency
geary.test(or$estimate, listw = m5)
```

```
##
##  Geary C test under randomisation
##
## data:  or$estimate
## weights: m5
##
## Geary C statistic standard deviate = 0.74408, p-value = 0.2284
## alternative hypothesis: Expectation greater than statistic
## sample estimates:
## Geary C statistic       Expectation          Variance
##      0.9768308399      1.0000000000      0.0009695831
```

Note that the value of Geary's C equal to 1 implies absence of spatial autocorrelation and a lower value of C between 0 and 1 represents a positive spatial autocorrelation. As the value approaches zero, it means strong autocorrelation. A high value greater than 1 represents negative spatial autocorrelation. If so, based on the above values, the value of C gets bigger as the order of adjacency matrices increases. This implies that the spatial autocorrelation decreases as the order of adjacency increases. This makes sense and aligns with our results thusfar because as you go further away from a point, you get less and less autocorrelation. Also, we see that as the order of adjacency matrices increase, the p-value gets larger and are larger than 0.001 for order 3, 4, and 5. This means that if confidence interval is 98%, distribution of estimate values in the counties are randomly distributed for higher order. Also, the c statistics value is smaller than 1, the spatial pattern is clustered, however, the values gets bigger as order increases implying that this effect weakens as order of adjancency increases.