library(sf)  
library(spdep)  
library(tidyverse)  
library(tmap)  
  
  
NC <- read\_sf("data/NC\_REGION.shp")  
  
NC\_UTM <- st\_transform(  
 x = NC,  
 crs = st\_crs("EPSG:26917")  
)  
queen\_nb <- NC\_UTM %>%  
 tibble::column\_to\_rownames("NAME") %>%  
 st\_as\_sf() %>%  
 poly2nb(queen = TRUE)  
  
  
queen\_nb\_w <- nb2listw(  
 neighbours = queen\_nb,  
 style = "W",  
 zero.policy = TRUE  
)

## Question 1

**Calculate Moran’s I for the following variables in the dataset. List the Moran’s I values and p-values for each variable in a single summary table:**

* MNEM2000
* MNEM1990
* TOTJOB2000
* TOTJOB1990

**Provide your R code for the Moran’s I tests (not the results themselves – that’s what the table is for).**

# It's totally OK if students do this part manually; I just was curious to find  
# a way to specify a set of variables to input automatically and then get a  
# table as an output.  
  
  
vars <- c("MNEM2000", "MNEM1990", "TOTJOB2000", "TOTJOB1990", "POP2000", "POP1990")  
  
  
moran\_results <-  
 NC\_UTM %>%  
 st\_drop\_geometry() %>%  
 dplyr::select(any\_of(x = vars)) %>%  
 map(function(x) {  
 moran.test(  
 x = x,  
 listw = queen\_nb\_w,  
 zero.policy = TRUE,  
 alternative = "two.sided"  
 )[c("estimate", "p.value")]  
 })  
  
moran\_table <- moran\_results %>%  
 transpose() %>%  
 as\_tibble() %>%  
 unnest\_wider(estimate) %>%  
 mutate(Variable = vars, .before = `Moran I statistic`)  
  
knitr::kable(moran\_table)

| Variable | Moran I statistic | Expectation | Variance | p.value |
| --- | --- | --- | --- | --- |
| MNEM2000 | 0.3884704 | -0.010101 | 0.0040192 | 3.238771e-10 |
| MNEM1990 | 0.4283699 | -0.010101 | 0.0040993 | 7.468829e-12 |
| TOTJOB2000 | 0.1631958 | -0.010101 | 0.0035409 | 0.003587933 |
| TOTJOB1990 | 0.1642235 | -0.010101 | 0.0037208 | 0.004264945 |
| POP2000 | 0.1647821 | -0.010101 | 0.0035306 | 0.003248218 |
| POP1990 | 0.1642235 | -0.010101 | 0.0037208 | 0.004264945 |

## Question 2:

**Calculate Moran’s I for the MNEM2000 variable using four different versions of :**

* Queen’s Case
* Rook’s Case
* nearest neighbors
* Maximum Distance (100% threshold)

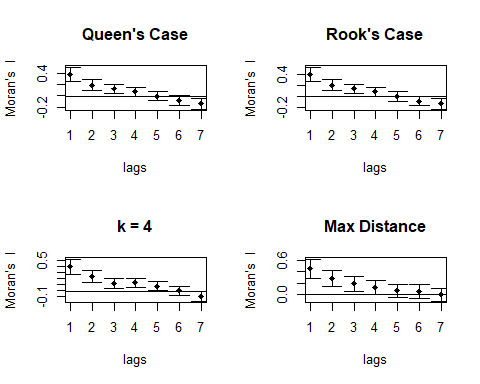
**Make sure your weights matrices are row standardized. Provide a table that summarized the calculated I, -values, and average number of connections per county. Discuss any systematic changes you observe in I, -values, and average links.**

NC\_centroids <- st\_centroid(NC\_UTM)  
  
q\_w <- NC\_UTM %>%  
 poly2nb(queen = TRUE)  
r\_w <- NC\_UTM %>%  
 poly2nb(queen = FALSE)  
  
k\_w <- knn2nb(knearneigh(NC\_centroids,  
 k = 4  
))  
  
max\_dist <- knn2nb(  
 knearneigh(  
 x = NC\_centroids,  
 k = 1  
 )  
) %>%  
 nbdists(coords = NC\_centroids$geometry) %>%  
 unlist() %>%  
 max()  
  
d\_w <- dnearneigh(  
 NC\_centroids,  
 d1 = 0,  
 d2 = max\_dist  
)  
  
list\_w <- list(  
 "Queen's Case" = q\_w,  
 "Rook's Case" = r\_w,  
 "k = 4" = k\_w,  
 "Max Distance" = d\_w  
)  
  
list\_w\_weighed <-  
 map(list\_w, ~ nb2listw(.x, style = "W", zero.policy = TRUE))  
  
moran\_w <- list\_w\_weighed %>%  
 map(function(w) {  
 moran.test(  
 x = NC\_UTM$MNEM2000,  
 listw = w,  
 zero.policy = TRUE,  
 alternative = "two.sided"  
 )[c("estimate", "p.value")]  
 })  
  
neighbors <- list\_w\_weighed %>%  
 map(function(w) {  
 w[["neighbours"]] %>%  
 unclass() %>%  
 map(length) %>%  
 unlist() %>%  
 mean()  
 }) %>%  
 enframe() %>%  
 rename(Links = value)  
  
moran\_table\_w <- moran\_w %>%  
 transpose() %>%  
 as\_tibble() %>%  
 unnest\_wider(estimate) %>%  
 mutate(Method = names(list\_w\_weighed), .before = `Moran I statistic`) %>%  
 bind\_cols(neighbors["Links"])  
  
knitr::kable(moran\_table\_w)

| Method | Moran I statistic | Expectation | Variance | p.value | Links |
| --- | --- | --- | --- | --- | --- |
| Queen’s Case | 0.3884704 | -0.010101 | 0.0040192 | 3.238771e-10 | 4.9 |
| Rook’s Case | 0.4044841 | -0.010101 | 0.0042276 | 1.814863e-10 | 4.62 |
| k = 4 | 0.4024844 | -0.010101 | 0.0041371 | 1.412727e-10 | 4 |
| Max Distance | 0.4536758 | -0.010101 | 0.0074840 | 8.278861e-08 | 2.8 |

## Question 3:

corr\_list <- list\_w %>%  
 map(~ sp.correlogram(  
 neighbours = .x,  
 var = NC\_UTM$MNEM2000,  
 order = 7,  
 method = "I",  
 zero.policy = TRUE  
 ))  
  
  
par(mfrow = c(2, 2))  
corr\_plots <- corr\_list %>%  
 imap(~ {  
 plot(.x, main = .y)  
 })

 *The general relationship is that the statistic decreases with the number of lags. However, in certain graphs, e.g. , there are small “bumps” where the value increases. This likely indicates that the “centers” of clusters are approximately 3~4 units (counties) away from one another.*

## Question 4:

**4. Adopt the code above to provide LISA maps for the MNEM1990 variable at an alpha of 0.1, 0.05, and 0.01 using the Queen’s case W. Provide a brief explanation of your findings about any local spatial autocorrelation at these more restrictive values of alpha.**

The map in this one is slightly different than the example in the “old” R Module 9. I’m using the quadrants generated by the function itself, which is still based on the mean. I’m not quite sure why the results seem to be different.

local\_moran\_queen <- localmoran(  
 x = NC\_UTM$MNEM2000,  
 listw = queen\_nb\_w,  
 # zero.policy = TRUE,  
 alternative = "two.sided"  
)  
  
  
lm\_pal <-  
 c(  
 "High-High" = "red",  
 "High-Low" = "#FF000080",  
 "Insignificant" = "gray90",  
 "Low-High" = "#0000FF80",  
 "Low-Low" = "blue"  
 )  
  
  
  
quadr <- attr(local\_moran\_queen, "quadr")[, 2]  
  
NC\_localmoran <- NC\_UTM  
  
NC\_localmoran$Quadrant <- quadr  
  
NC\_localmoran$p.value <- local\_moran\_queen[, 5]  
  
sig\_level <- 0.5  
  
plot\_LISA\_sig <- function(sig) {  
 NC\_localmoran <- NC\_localmoran %>%  
 mutate(  
 Quadrant\_Sig = if\_else(p.value <= sig,  
 true = as.character(Quadrant),  
 false = "Insignificant"  
 ) %>%  
 as.factor()  
 )  
 t <- tm\_shape(NC\_localmoran) +  
 tm\_polygons(  
 col = "Quadrant\_Sig",  
 palette = lm\_pal,  
 legend.is.portrait = F,  
 border.col = "black"  
 ) +  
 tm\_layout(  
 main.title = paste0("LISA Cluster at Alpha = ", sig),  
 legend.outside = T,  
 legend.outside.position = "bottom"  
 )  
  
 return(t)  
}  
  
LISA\_list <- map(c(0.5, 0.1, 0.05, 0.01), plot\_LISA\_sig)  
  
tmap\_arrange(LISA\_list)

