Local Indicators of Spatial Autocorrelation (LISA)

1. From previous exercises, you have a good idea of how to construct spatial weights and use those weights together with information on a variable of interest to generate a Moran plot and perform a Moran test for the existence of spatial autocorrelation in a study region as a whole.

2. Make sure that you have the following libraries loaded

```
>library(tmap)
>library(spdep)
>library(sf)
>library(sp)
```

3. Let us look at another variable from the sids shape file of North Carolina. This variable is SID79 and it maps the number of sudden infant deaths in 1979. The number of SIDS deaths is strongly influenced by the birth rate (BIR79). So we standardize the SID79 value by BIR79 to get the rate of SIDS deaths. This is the variable SIDSRT79. Set your current working directory and load the sids.shp file again

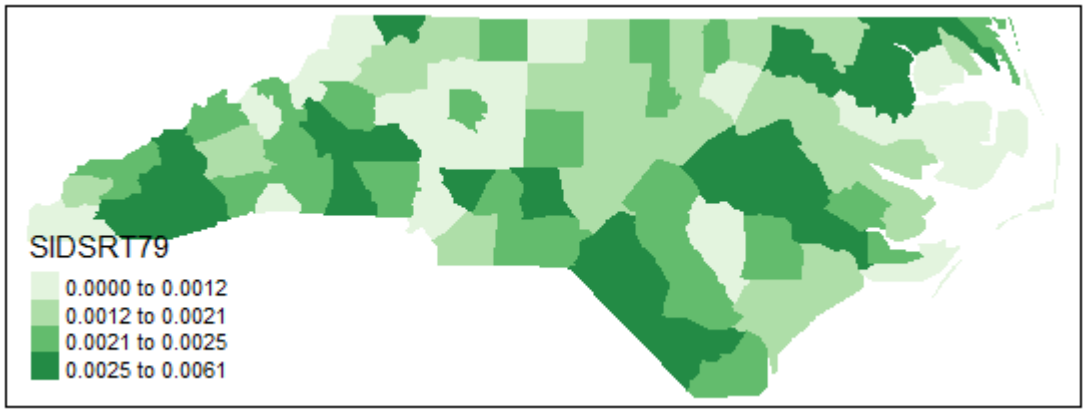
```
> ncmap <- st_read("sids.shp")ncmap
```

4. Make sure we have a spatial object

```
> ncmap2 <- as(ncmap, “Spatial”)
```

5. Let’s have a quick look at the map

```
> tm_shape(ncmap2) + tm_fill("SIDSRT79", style="quantile", n=4, palette="Greens")
```



6. Now we create some rook neighbors for our new shape file data

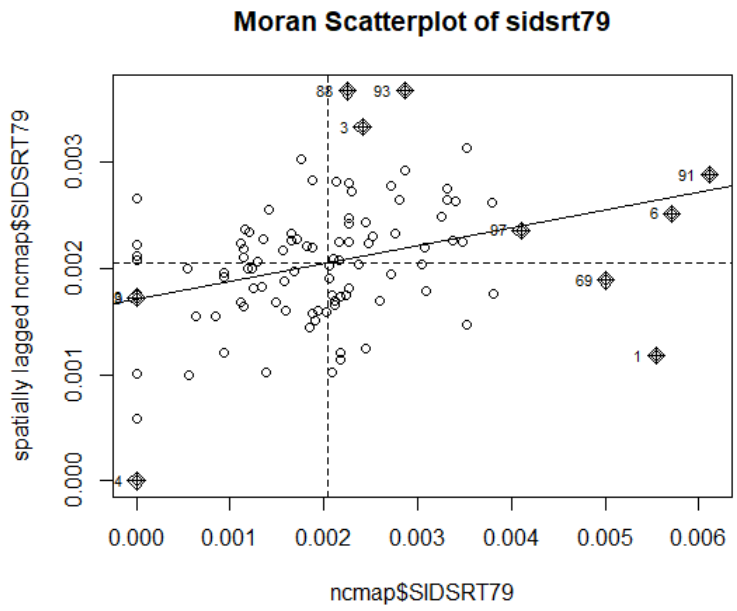
```
> ncmap2rook <- poly2nb(ncmap2, queen=FALSE)
> ncmap2rook
```

Neighbour list object:
Number of regions: 100
Number of nonzero links: 462
Percentage nonzero weights: 4.62

Average number of links: 4.62

And convert the neighbor links into a listw object
> ncmap2rlist <- nb2listw(ncmap2rook)

7. Let's run a quick Moran plot to look at the overall pattern of spatial autocorrelation for the sidsrt79 variable our using rook contiguous neighbors. I am being lazy here and not standardizing…you don't have to standardize though that is common in the literature and helps comparing patterns.
> moran.plot(ncmap2$SIDSRT79, listw=ncmap2rlist, main=c("Moran Scatterplot of sidsrt79"))



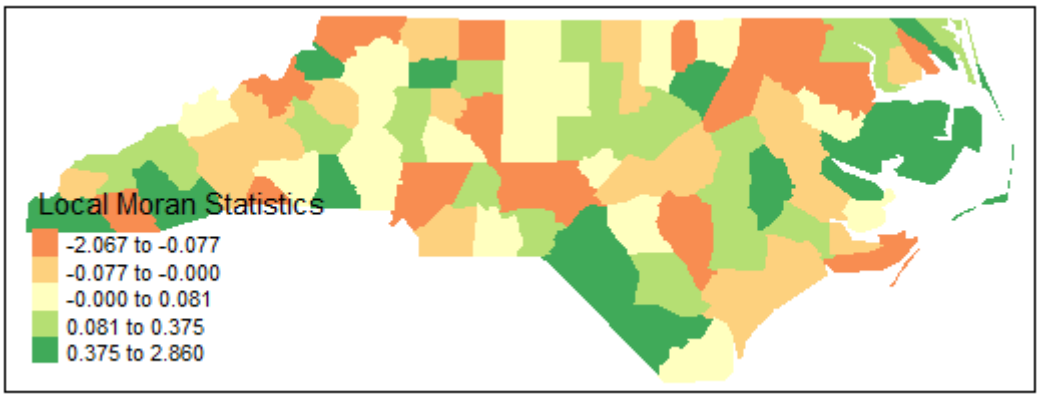
The scatterplot is still broadly positive, though not as strongly so as the plot from Exercise 13.

8. Now let's think about the contribution of our individual areas or counties to the Moran scatterplot. How does the strength of spatial autocorrelation in the variable sidsrt79 vary across the counties in North Carolina? We can examine this using the localmoran() function. Note in the code below I made sure that our rook weights are row standardized
> local <- localmoran(x = ncmap2$SIDSRT79, listw=nb2listw(ncmap2rook, style="W"))

The localmoran() function generates a variety of different kinds of outputs, check with ?localmoran. Useful statistics from the localmoran model include
- observed local Moran statistics
- expected values of local Moran statistics and their variance
- standard deviates and p-values

9. Let's map the local Moran statistics. These are values of Moran's I generated for each county in the shapefile. To do this, we have to bind results to our shapefile

```
> lmoran_map <- cbind(ncmap2, local)
> tm_shape(lmoran_map) +tm_fill(col="Ii", style="quantile", title=" Local Moran Statistics")
```

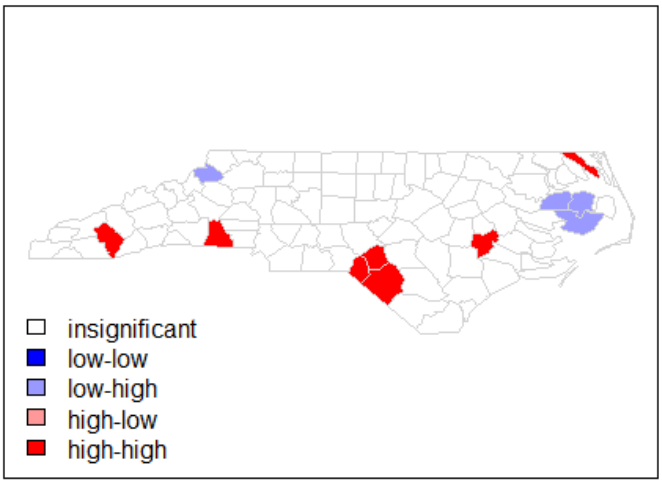


The map highlights how spatial autocorrelation variables change over local areas in North Carolina. What would be even better would be if we could detect clusters of counties with high and low values of sidsrt79.

NB From this point, I would just have a quick read and not worry about the coding, for it is messy and GeoDa in the next exercise will make everything so much easier!

10. We can search for hot spots (regions of high or low value), but the coding is a bit of a mess.

```
> quadrant <- vector(mode="numeric",length=nrow(local))
> m.sidsrt79 <- ncmap2$SIDSRT79 - mean(ncmap2$SIDSRT79)
> m.local <- local[,1] - mean(local[,1])
> signif <- 0.1
> quadrant[m.sidsrt79 >0 & m.local>0] <- 4
> quadrant[m.sidsrt79 <0 & m.local<0] <- 1
> quadrant[m.sidsrt79 <0 & m.local>0] <- 2
> quadrant[m.sidsrt79 >0 & m.local<0] <- 3
> quadrant[local[,5]>signif] <- 0
> brks <- c(0,1,2,3,4)
> colors <- c("white","blue",rgb(0,0,1,alpha=0.4),rgb(1,0,0,alpha=0.4),"red")
> plot(ncmap2,border="lightgray",col=colors[findInterval(quadrant,brks,all.inside=FALSE)])
> box()
> legend("bottomleft",legend=c("insignificant","low-low","low-high","high-low","high-high"),fill=colors,bty="n")
```



So what we have done here is group counties in the ways indicated in the legend.
The deep reds are counties with values of SIDSRT79 that are significantly greater than the average and which are surrounded by other counties that have SIDRT79 values that are significantly above average…..these are positive hot spots to focus attention on.

You should be able to work out the other patterns from the legend.

Getis-Ord Gi-statistic

11. Hot-spot analysis is also undertaken using the Getis-Ord Gi statistics. The Gi statistic looks at neighbors of a particular county to identify spatial clusters of high and low values of a variable of interest. Neighbors for a county are defined as those that have centroids within a certain distance of the focus county's centroid. (The Gi* statistic takes into account the value of the variable of interest at the given location, as well as the value of that variable in neighboring counties.)

12. So now we have to generate a set of distance weights to identify the neighbors of each county. Based on the discussion from Exercise 12, we first identify the coordinates in the ncmap object, then we set the upper limit on neighbors to extend to a distance of 75kms, and the third line of code makes a list object from the weights. Style = B for basic binary (0/1).

```
> coords <- coordinates(ncmap2)
> ncwm_d75 <- dnearneigh(coords, 0, 75, longlat=TRUE)
> ncwmlist_d75<- nb2listw(ncwm_d75, style="B", zero.policy=TRUE)
```

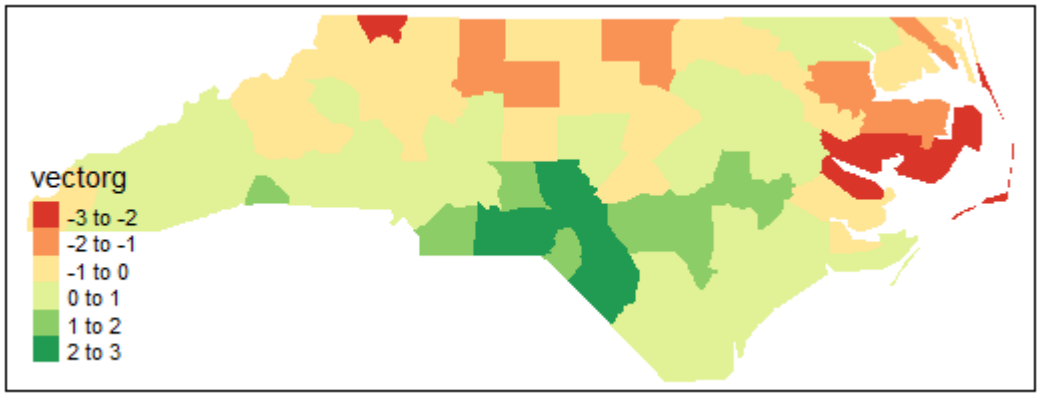
13. To generate the Gi statistic for each county in our North Carolina map, use the function localG() which is also part of the spdep package

```
> localg <- localG(ncmap2$SIDSRT79, ncwmlist_d75)
```

This command returns a localG object containing z-scores for the Gi statistic. Large positive z-scores indicate a cluster of above average SIDS and large negative values indicate clusters of below average SIDS.

14. It turns out that getting the object localg into some form that you can map is very cumbersome. I have not found an elegant solution, but the following sort of works

```
> vectorg <- (localg[1:100])
> as.numeric(vectorg)
> NAME <- ncmap2$NAME
> newvar <- data.frame(NAME, vectorg)
> ncmap3 <-merge(ncmap2, newvar, by = "NAME")
> tm_shape(ncmap3) +tm_fill("vectorg")
```



Note that the resulting map is not that different from the LISA map.

You might try this with different weighting methods…..but we can do this very easily using GEODA (see Exercise 15)