Spatial Weights Exercise

In this exercise we will primarily use the R package spdep to generate spatial weights, or the types of linkages between regions that exist in our study regions. We will also use a couple of other packages to add to your repertoire.

1. So please install the following packages, and choose to install any dependencies in the process:
sp
spdep

So,

```
> install.packages("spdep", dependencies=TRUE)
> install.packages("sp", dependencies=TRUE)
```

*You may need >sf_use_s2(FALSE)

Note that spdep was written to work with spatial objects and so a couple of extra steps may be needed when using the sf library, sp and spdep together.

2. Load the following libraries: tmap, sf, sp, spdep and set your working directory.

3. You will be working with a North Carolina county shapefile named sids.shp. The sids zip folder can be found under the Datasets folder for Module 4. You need to download the folder and unzip its contents for use. Then make sure you set your current working directory in R to make reading your input data easier.

4. Read the shape file using st_read

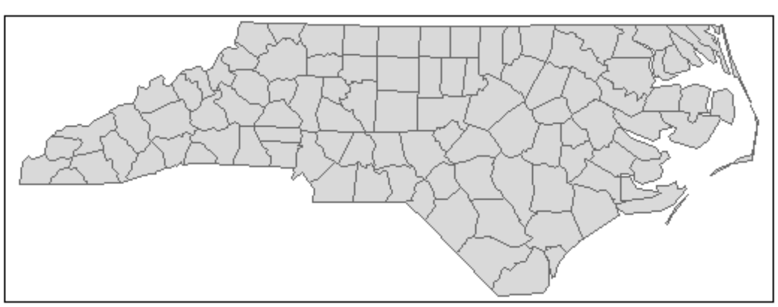
```
> ncmap <- st_read("sids.shp")
```

5. To make sure that spdep reads the ncmap data as spatial, let's turn it into a spatial object

```
> ncmap2 <- as(ncmap, "Spatial")
```

6. As usual you can use the qtm() function to generate a quick map

```
> qtm(ncmap2)
```



7. Now use the polygon to neighbors function from spdep to generate a set of queen contiguity neighbors for each county in North Carolina and then look at the new object with the summary function that provides some useful information including the total number of links etc.

```
> ncwm_q <-poly2nb(ncmap2, queen=TRUE)
> summary(ncwm_q)
```

```
Neighbour list object:
Number of regions: 100
Number of nonzero links: 490
Percentage nonzero weights: 4.9
Average number of links: 4.9
Link number distribution:
 2  3  4  5  6  7  8  9
 8 15 17 23 19 14  2  2
8 least connected regions:
3 20 44 55 76 79 89 98 with 2 links
2 most connected regions:
38 66 with 9 links
```

8. You can examine the neighbors of a particular county as follows

```
> ncwm_q[[1]]
[1]  2 18 19
```

And so the county listed #1 has neighbor counties: 2, 18 & 19.
Which are these counties?

```
> ncmap2$NAME[1]
[1] "Ashe"
```

And the neighbors of Ashe county are?

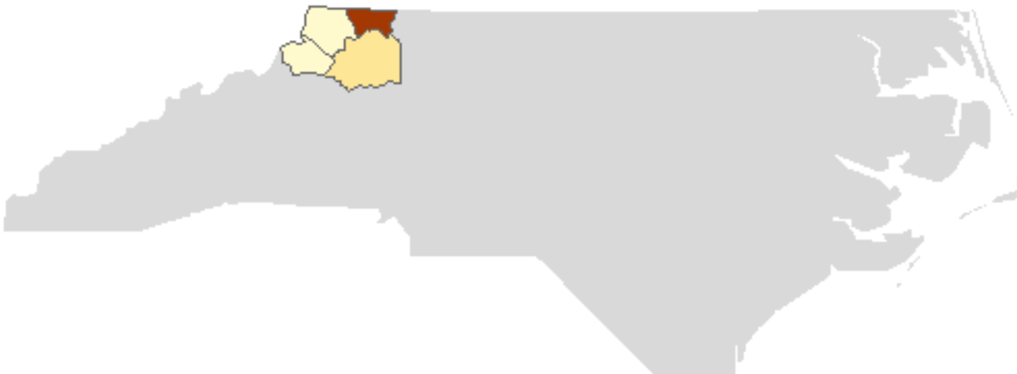
```
> ncmap2$NAME[c(2,18,19)]
[1] "Alleghany" "Wilkes"    "Watauga"
```

9. Just to check everything is working the way we want, let's map this little neighborhood of counties in tmap. Now, it is important to note that the observations numbers used by spdep refer simply to the rows in the data file, not to any of the variables noted. However, we have the names of the counties that are queen neighbors of Ashe county, so we can use the variable ncmap$NAME to map them. First let us subset the counties we want and then map them:

```
> ncmap2_asheq <-subset(ncmap2, ncmap2$NAME=="Ashe" | ncmap2$NAME=="Alleghany" | ncmap2$NAME=="Wilkes" | ncmap2$NAME=="Watauga",)
```

```
> tm_shape(ncmap2) +tm_fill() + tm_shape(ncmap2_asheq) + tm_fill("SIDSRT79") +tm_legend(show=FALSE) +tm_borders()
```

And you should see the following (the county which is Ashe should be obvious to you)



10. We can find rook contiguous neighbors as
> ncwm_r <- poly2nb(ncmap2, queen=FALSE)
> summary(ncwm_r)
Neighbour list object:
Number of regions: 100
Number of nonzero links: 462
Percentage nonzero weights: 4.62
Average number of links: 4.62
Link number distribution:

```
 2  3  4  5  6  7  8  9
 8 18 20 25 21  4  3  1
```

8 least connected regions:
3 20 44 55 76 79 89 98 with 2 links
1 most connected region:
38 with 9 links
Notice that there are slightly fewer rook contiguous neighbors as queen contiguous neighbors as we might expect.

11. We can plot our neighbor maps, first with queen contiguity, then with rook contiguity, & with both (check the queen contiguity again for Ashe county)

> plot(ncmap2, border = 'lightgrey')

> plot(ncwm_q, coordinates(ncmap2), add=TRUE, col='red')

> plot(ncmap2, border = 'lightgrey')
> plot(ncwm_r, coordinates(ncmap2), add=TRUE, col='blue')

And overlaying these maps, so you can see the links added by the queen contiguity
> plot(ncmap2, border = 'lightgrey')
> plot(ncwm_q, coordinates(ncmap2), add=TRUE, col='red')
> plot(ncwm_r, coordinates(ncmap2), add=TRUE, col='blue')

I count 14 red lines on the map x2 = 28, the difference in the number of links reported above.

12. Let us now use spdep to find distance-based neighbors of each county. Here we use the dnearneigh() function. The function finds the neighbors of each area based on distances between area centroids. We have to supply a lower limit and an upper limit of distances for the dnearneigh() function. We can set the lower limit at 0, but we need to set the upper limit for the function at a value that is large enough to ensure that each of our counties has at least one neighbor. That means finding the county which is most remote in terms of distance.

First we find the coordinates for our map areas

```
> coords <- coordinates(ncmap2)
> head(coords)
       [,1]     [,2]
0 -81.49826 36.43140
1 -81.12515 36.49101
2 -80.68575 36.41252
3 -76.05211 36.39714
4 -77.41056 36.42228
5 -76.99478 36.36145
```

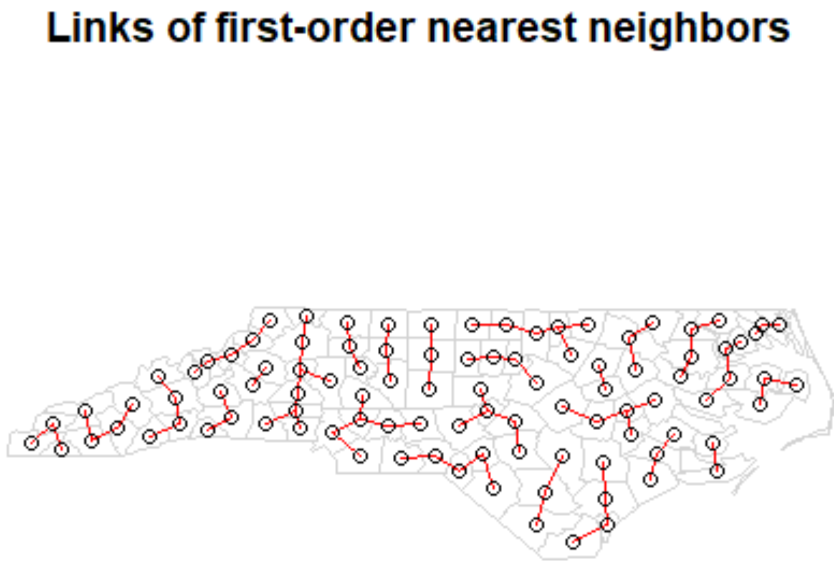
Then we find the nearest neighbor to each county

```
> k1 <- knn2nb(knearneigh(coords))
> k1dists <-unlist(nbdists(k1, coords, longlat=TRUE))
> summary(k1dists)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  12.21   27.37   29.84   29.89   33.18   45.34 
```

So the largest (first-order) nearest neighbor distance = 45.34, so we can set the upper bound for neighbors at 46.

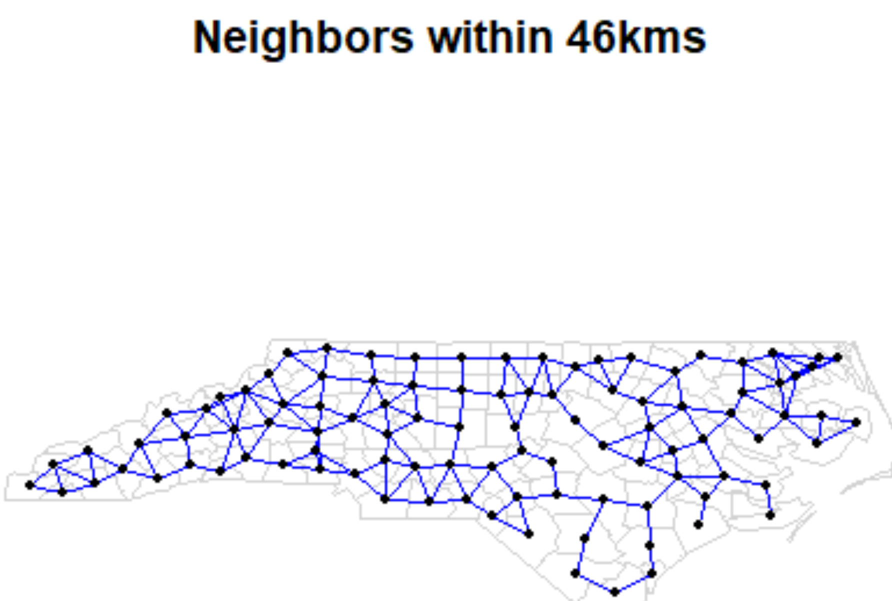
13.Let's look at the nearest neighbors for each county

```
> plot(ncmap2, border = 'lightgrey')
> plot(k1, coords, add=TRUE, col="red")
> title(main = "Links of first-order nearest neighbors", cex = 0.6)
```



13. And now plot the number of neighbors within a distance band of 46kms
First, we identify all neighbors of areas within a distance of 0-46kms and then plot

```
> ncwm_d46 <- dnearneigh(coords, 0, 46, longlat=TRUE)
> plot(ncmap2, border = 'lightgrey')
> plot(ncwm_d46, coords, add=TRUE, col="blue", pch=19, cex=0.6)
> title(main="Neighbors within 46kms", cex=0.6)
```



You can adjust the code above to look at k=1, 2,…, n nearest neighbors.

14. And we can find the full set of inverse distance weights between all pairs of counties using the nbdists() function of spdep. The function requires a neighbor list and we need that list to include all

possible neighbors, so first we use the dnearneigh() function again with a very large upper bound to capture all links

```
> ncwm_d1000 <- dnearneigh(coords, 0, 1000, longlat=TRUE)
> ncwm_d1000
Neighbour list object:
Number of regions: 100
Number of nonzero links: 9900
Percentage nonzero weights: 99
Average number of links: 99
```

And then we can find the inverse distance weights as

```
> dist <- nbdists(ncwm_d1000, coords, longlat=TRUE)
> idw <- lapply(dist, function(x) 1/(x))
> idw[1]
[[1]]
 [1] 0.029332006 0.013717995 0.002047444 0.002728219 0.002474962
 [6] 0.002118373 0.002326764 0.003290959 0.008817162 0.005152260
[11] 0.006450682 0.003914749 0.004420849 0.003609755 0.002894785
[16] 0.002149129 0.025198008 0.034795573 0.002204753 0.002274730
[21] 0.018145498 0.012349669 0.003433485 0.008555571 0.006280441
[26] 0.005160067 0.002452742 0.004593781 0.004170984 0.003124316
[31] 0.013125572 0.002811727 0.018553545 0.010609348 0.002495992
[36] 0.003758497 0.007926188 0.011143435 0.009728990 0.015570283
[41] 0.007341258 0.012761717 0.002234657 0.002082824 0.010279180
[46] 0.005793856 0.004591580 0.003010383 0.008017687 0.002610973
[51] 0.011165592 0.007671637 0.003330660 0.006040323 0.001950727
[56] 0.002323078 0.004799928 0.002775777 0.004251211 0.008286412
[61] 0.002967922 0.003773923 0.008173879 0.009222440 0.005115026
[66] 0.004524007 0.006903457 0.006934364 0.005287795 0.005965900
[71] 0.006651055 0.004122777 0.002683730 0.005554490 0.007709510
[76] 0.007048422 0.004444623 0.003074120 0.002217042 0.003665159
[81] 0.003489907 0.002462068 0.005474742 0.004878177 0.003871818
[86] 0.002070861 0.002754912 0.004453766 0.003930102 0.002424696
[91] 0.003931351 0.002423204 0.003394041 0.002131498 0.002989748
[96] 0.002579512 0.002832470 0.002460765 0.002535437
```

So these are 99 inverse distance values based on the inverse of the distances between each county and all its 99 neighboring counties in North Carolina.

15. To this point, we have found our neighbors using different techniques. The next step is to assign weights to those neighbors for use in subsequent analysis. We did this using inverse distance weights based on distances between area centroids. But, what about building weights for contiguity neighbors and our k nearest neighbor functions? Let us take the example of our queen contiguity

first-order weights generated above. Just to refresh your memory, here are the queen contiguity neighbors of the first six counties

```
> head(ncwm_q)
[[1]]
[1]  2 18 19
[[2]]
[1]  1  3 18
[[3]]
[1]  2 10 18 23 25
[[4]]
[1]  7 56
[[5]]
[1]  6  9 16 28
[[6]]
[1]  5  8 28
```

We use the nb2listw() function in spdep to generate weights for each neighbor. These weights tell us how much each of the neighbors might influence the focus area or county in this example. So here we create weights for our queen contiguity neighbors

```
> rsncwm_q <- nb2listw(ncwm_q, style="W", zero.policy=TRUE)
```

The style = "W" option indicates to R that we are building row standardized weights (where the weights for each county sum to 1). Row standardizing weights allows for comparability between areas with different numbers of neighbors. The zero.policy=TRUE option tells R not to worry about counties with no neighbors, if they exist.

For the queen contiguity neighbors of each county, the row standardized weights are in the object rsncwm_q. You can see the weights below

```
> head(rsncwm_q)
$style
[1] "W"

$neighbours
Neighbour list object:
Number of regions: 100
Number of nonzero links: 490
Percentage nonzero weights: 4.9
Average number of links: 4.9

$weights
$weights[[1]]
[1] 0.3333333 0.3333333 0.3333333
```

```
$weights[[2]]
[1] 0.3333333 0.3333333 0.3333333

$weights[[3]]
[1] 0.2 0.2 0.2 0.2 0.2

$weights[[4]]
[1] 0.5 0.5

$weights[[5]]
[1] 0.25 0.25 0.25 0.25

$weights[[6]]
[1] 0.3333333 0.3333333 0.3333333
```