*Spatial Statistics Lab 5*

**Onyedikachi J Okeke**

2024-02-17

### 0.0 To Load the library

library(sp)  
library(spatstat)  
library(sf)  
library(spatstat.geom)  
library(ctv)  
library(terra)  
library(spdep)

##### Library

## Loading required package: spatstat.data

## Loading required package: spatstat.geom

## spatstat.geom 3.2-8

## Loading required package: spatstat.random

## spatstat.random 3.2-2

## Loading required package: spatstat.explore

## Loading required package: nlme

## spatstat.explore 3.2-5

## Loading required package: spatstat.model

## Loading required package: rpart

## spatstat.model 3.2-8

## Loading required package: spatstat.linnet

## spatstat.linnet 3.1-3

##   
## spatstat 3.0-7   
## For an introduction to spatstat, type 'beginner'

## Linking to GEOS 3.11.2, GDAL 3.7.2, PROJ 9.3.0; sf\_use\_s2() is TRUE

## terra 1.7.65

##   
## Attaching package: 'terra'

## The following objects are masked from 'package:spatstat.geom':  
##   
## area, delaunay, is.empty, rescale, rotate, shift, where.max,  
## where.min

## Loading required package: spData

### Q1 Loading Sids data

# Define the file path  
file\_path <- "C:/Spatial Statistics Labwork/Lab5Data/sids2.shp"  
  
# Import the shapefile  
sids <- st\_read(file\_path)  
  
class(sids)  
  
# Check the structure of the imported object  
str(sids)

##### Result

## Reading layer `sids2' from data source   
## `C:\Spatial Statistics Labwork\Lab5Data\sids2.shp' using driver `ESRI Shapefile'  
## Simple feature collection with 100 features and 18 fields  
## Geometry type: MULTIPOLYGON  
## Dimension: XY  
## Bounding box: xmin: -84.32385 ymin: 33.88199 xmax: -75.45698 ymax: 36.58965  
## CRS: NA

## [1] "sf" "data.frame"

## Classes 'sf' and 'data.frame': 100 obs. of 19 variables:  
## $ AREA : num 0.114 0.061 0.143 0.07 0.153 0.097 0.062 0.091 0.118 0.124 ...  
## $ PERIMETER: num 1.44 1.23 1.63 2.97 2.21 ...  
## $ CNTY\_ : num 1825 1827 1828 1831 1832 ...  
## $ CNTY\_ID : num 1825 1827 1828 1831 1832 ...  
## $ NAME : chr "Ashe" "Alleghany" "Surry" "Currituck" ...  
## $ FIPS : chr "37009" "37005" "37171" "37053" ...  
## $ FIPSNO : num 37009 37005 37171 37053 37131 ...  
## $ CRESS\_ID : int 5 3 86 27 66 46 15 37 93 85 ...  
## $ BIR74 : num 1091 487 3188 508 1421 ...  
## $ SID74 : num 1 0 5 1 9 7 0 0 4 1 ...  
## $ NWBIR74 : num 10 10 208 123 1066 ...  
## $ BIR79 : num 1364 542 3616 830 1606 ...  
## $ SID79 : num 0 3 6 2 3 5 2 2 2 5 ...  
## $ NWBIR79 : num 19 12 260 145 1197 ...  
## $ SIDR74 : num 0.917 0 1.568 1.969 6.334 ...  
## $ SIDR79 : num 0 5.54 1.66 2.41 1.87 ...  
## $ NWR74 : num 9.17 20.53 65.24 242.13 750.18 ...  
## $ NWR79 : num 13.9 22.1 71.9 174.7 745.3 ...  
## $ geometry :sfc\_MULTIPOLYGON of length 100; first list element: List of 1  
## ..$ :List of 1  
## .. ..$ : num [1:27, 1:2] -81.5 -81.5 -81.6 -81.6 -81.7 ...  
## ..- attr(\*, "class")= chr [1:3] "XY" "MULTIPOLYGON" "sfg"  
## - attr(\*, "sf\_column")= chr "geometry"  
## - attr(\*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA NA NA NA NA NA NA ...  
## ..- attr(\*, "names")= chr [1:18] "AREA" "PERIMETER" "CNTY\_" "CNTY\_ID" ...

### Check the projection of the shapefile

# Extract CRS information using st\_crs()  
crs\_info <- st\_crs(sids)  
  
# Print CRS information  
print(crs\_info)

##### Result

## Coordinate Reference System: NA

### Define the target CRS (NAD27 UTM Zone 17N

# Assign the coordinate system (WGS84)  
st\_crs(sids) <- st\_crs("+proj=longlat +ellps=WGS84")  
  
# Check the current CRS  
print(st\_crs(sids))  
  
# Define the target CRS (NAD27 UTM Zone 17N)  
target\_crs <- "+proj=utm +zone=17 +datum=NAD27"  
  
# Project the shapefile to the target CRS  
sids\_projected <- st\_transform(sids, target\_crs)  
  
# Check the CRS of the projected shapefile  
print(st\_crs(sids\_projected))

##### Result

## Coordinate Reference System:  
## User input: +proj=longlat +ellps=WGS84   
## wkt:  
## GEOGCRS["unknown",  
## DATUM["Unknown based on WGS 84 ellipsoid",  
## ELLIPSOID["WGS 84",6378137,298.257223563,  
## LENGTHUNIT["metre",1],  
## ID["EPSG",7030]]],  
## PRIMEM["Greenwich",0,  
## ANGLEUNIT["degree",0.0174532925199433],  
## ID["EPSG",8901]],  
## CS[ellipsoidal,2],  
## AXIS["longitude",east,  
## ORDER[1],  
## ANGLEUNIT["degree",0.0174532925199433,  
## ID["EPSG",9122]]],  
## AXIS["latitude",north,  
## ORDER[2],  
## ANGLEUNIT["degree",0.0174532925199433,  
## ID["EPSG",9122]]]]

## Coordinate Reference System:  
## User input: +proj=utm +zone=17 +datum=NAD27   
## wkt:  
## PROJCRS["unknown",  
## BASEGEOGCRS["unknown",  
## DATUM["North American Datum 1927",  
## ELLIPSOID["Clarke 1866",6378206.4,294.978698213898,  
## LENGTHUNIT["metre",1]],  
## ID["EPSG",6267]],  
## PRIMEM["Greenwich",0,  
## ANGLEUNIT["degree",0.0174532925199433],  
## ID["EPSG",8901]]],  
## CONVERSION["UTM zone 17N",  
## METHOD["Transverse Mercator",  
## ID["EPSG",9807]],  
## PARAMETER["Latitude of natural origin",0,  
## ANGLEUNIT["degree",0.0174532925199433],  
## ID["EPSG",8801]],  
## PARAMETER["Longitude of natural origin",-81,  
## ANGLEUNIT["degree",0.0174532925199433],  
## ID["EPSG",8802]],  
## PARAMETER["Scale factor at natural origin",0.9996,  
## SCALEUNIT["unity",1],  
## ID["EPSG",8805]],  
## PARAMETER["False easting",500000,  
## LENGTHUNIT["metre",1],  
## ID["EPSG",8806]],  
## PARAMETER["False northing",0,  
## LENGTHUNIT["metre",1],  
## ID["EPSG",8807]],  
## ID["EPSG",16017]],  
## CS[Cartesian,2],  
## AXIS["(E)",east,  
## ORDER[1],  
## LENGTHUNIT["metre",1,  
## ID["EPSG",9001]]],  
## AXIS["(N)",north,  
## ORDER[2],  
## LENGTHUNIT["metre",1,  
## ID["EPSG",9001]]]]

### Define the CRS for North Carolina NAD 83 State Plane

# Define the CRS for North Carolina NAD 83 State Plane  
nc\_crs <- st\_crs("+proj=lcc +lat\_1=34.33333333333334 +lat\_2=36.16666666666666 +lat\_0=33.75 +lon\_0=-79 +x\_0=609601.22 +y\_0=0 +ellps=GRS80 +datum=NAD83 +units=m +no\_defs")  
  
# Transform the sids shapefile to North Carolina NAD 83 State Plane  
sids\_SP <- st\_transform(sids, nc\_crs)  
print(st\_crs(sids\_SP))

##### Result

## Coordinate Reference System:  
## User input: +proj=lcc +lat\_1=34.33333333333334 +lat\_2=36.16666666666666 +lat\_0=33.75 +lon\_0=-79 +x\_0=609601.22 +y\_0=0 +ellps=GRS80 +datum=NAD83 +units=m +no\_defs   
## wkt:  
## PROJCRS["unknown",  
## BASEGEOGCRS["unknown",  
## DATUM["North American Datum 1983",  
## ELLIPSOID["GRS 1980",6378137,298.257222101,  
## LENGTHUNIT["metre",1]],  
## ID["EPSG",6269]],  
## PRIMEM["Greenwich",0,  
## ANGLEUNIT["degree",0.0174532925199433],  
## ID["EPSG",8901]]],  
## CONVERSION["unknown",  
## METHOD["Lambert Conic Conformal (2SP)",  
## ID["EPSG",9802]],  
## PARAMETER["Latitude of false origin",33.75,  
## ANGLEUNIT["degree",0.0174532925199433],  
## ID["EPSG",8821]],  
## PARAMETER["Longitude of false origin",-79,  
## ANGLEUNIT["degree",0.0174532925199433],  
## ID["EPSG",8822]],  
## PARAMETER["Latitude of 1st standard parallel",34.3333333333333,  
## ANGLEUNIT["degree",0.0174532925199433],  
## ID["EPSG",8823]],  
## PARAMETER["Latitude of 2nd standard parallel",36.1666666666667,  
## ANGLEUNIT["degree",0.0174532925199433],  
## ID["EPSG",8824]],  
## PARAMETER["Easting at false origin",609601.22,  
## LENGTHUNIT["metre",1],  
## ID["EPSG",8826]],  
## PARAMETER["Northing at false origin",0,  
## LENGTHUNIT["metre",1],  
## ID["EPSG",8827]]],  
## CS[Cartesian,2],  
## AXIS["(E)",east,  
## ORDER[1],  
## LENGTHUNIT["metre",1,  
## ID["EPSG",9001]]],  
## AXIS["(N)",north,  
## ORDER[2],  
## LENGTHUNIT["metre",1,  
## ID["EPSG",9001]]]]

### Plotting the data in three different coordinate systems

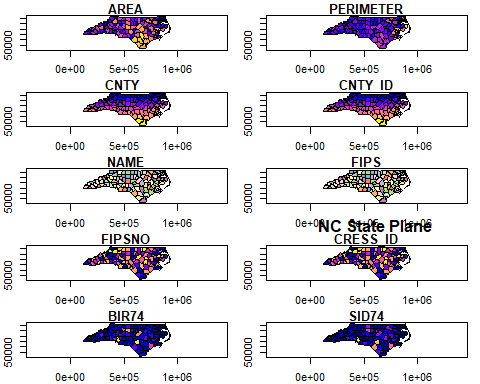
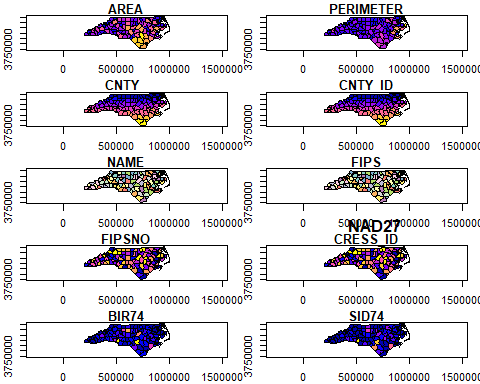
par(mfrow=c(2,2))  
  
plot(sids, axes=T)  
title("WGS84")  
  
plot(sids\_NAD, axes = TRUE)  
title("NAD27")  
  
plot(sids\_SP, axes=T)  
title("NC State Plane")

##### Result

## Warning: plotting the first 10 out of 18 attributes; use max.plot = 18 to plot  
## all  
  
## Warning: plotting the first 10 out of 18 attributes; use max.plot = 18 to plot  
## all  
  
## Warning: plotting the first 10 out of 18 attributes; use max.plot = 18 to plot  
## all

**Q2**  
A group of maps with different colored dots

Description automatically generated



### Contiguity based neighbors

# Create Queen contiguity spatial weights matrix  
sids\_nbq <- poly2nb(sids, queen = TRUE)  
# Check the summary of the spatial weights matrix  
summary(sids\_nbq)  
  
# Create Queen (false) contiguity spatial weights matrix  
sids\_nbr <- poly2nb(sids, queen = FALSE)  
# Check the summary of the spatial weights matrix  
summary(sids\_nbr)  
  
plot(st\_geometry(sids), main="Spatial Plot with Neighborhoods")  
  
coords <- st\_coordinates(sids)  
  
# Check the number of neighborhoods in sids\_nbq  
num\_neighborhoods <- length(sids\_nbq)  
  
# Check the number of coordinate pairs in coords  
num\_coords <- nrow(coords)  
  
# Print out the counts for verification  
print(paste("Number of neighborhoods:", num\_neighborhoods))  
print(paste("Number of coordinate pairs:", num\_coords))  
  
# Example adjustment (assuming you need to align coords\_2d with sids\_nbq correctly)  
coords\_2d\_corrected <- coords[1:num\_neighborhoods, ]  
  
par(mfrow=c(1,1))  
  
plot(sids\_nbq, coords\_2d\_corrected)  
plot(sids, add= TRUE)  
  
plot(sids\_nbr, coords\_2d\_corrected)  
plot(sids, add= TRUE)

##### Result

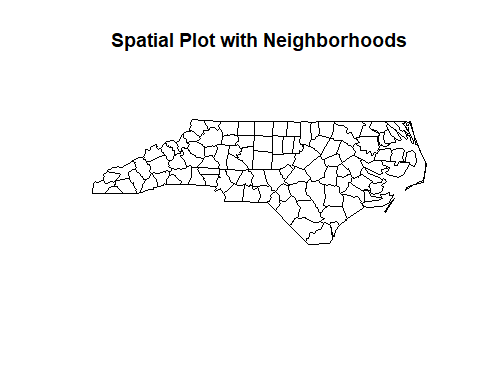
## Neighbour list object:  
## Number of regions: 100   
## Number of nonzero links: 490   
## Percentage nonzero weights: 4.9   
## Average number of links: 4.9   
## Link number distribution:  
##   
## 2 3 4 5 6 7 8 9   
## 8 15 17 23 19 14 2 2   
## 8 least connected regions:  
## 4 21 45 56 77 80 90 99 with 2 links  
## 2 most connected regions:  
## 39 67 with 9 links

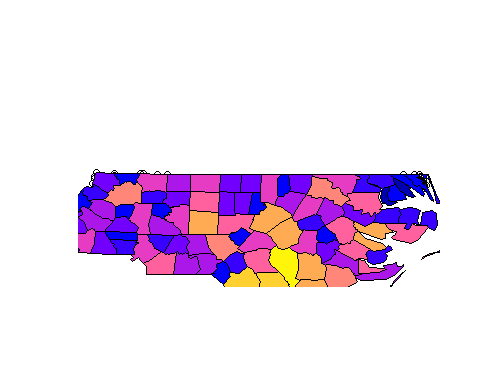
## Neighbour list object:  
## Number of regions: 100   
## Number of nonzero links: 462   
## Percentage nonzero weights: 4.62   
## Average number of links: 4.62   
## Link number distribution:  
##   
## 2 3 4 5 6 7 8 9   
## 8 18 20 25 21 4 3 1   
## 8 least connected regions:  
## 4 21 45 56 77 80 90 99 with 2 links  
## 1 most connected region:  
## 39 with 9 links

## [1] "Number of neighborhoods: 100"

## [1] "Number of coordinate pairs: 2529"

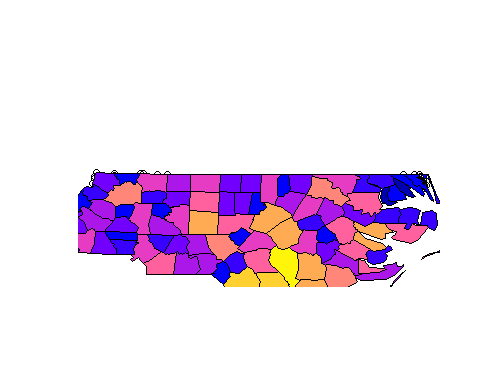
## Warning in plot.sf(sids, add = TRUE): ignoring all but the first attribute  
  
## Warning in plot.sf(sids, add = TRUE): ignoring all but the first attribute





### Q3 Plot the ROOK case and take a screenshot of your figure. Briefly describe the difference between the figure from Q2 and Q3

The difference between ROOK and QUEEN contiguity methods in spatial analysis is highlighted by how they define neighboring regions. ROOK contiguity considers regions as neighbors only if they share an edge, leading to a somewhat stricter and potentially sparser neighborhood network. In contrast, QUEEN contiguity expands this definition to include regions that share at least a vertex (corner point), in addition to those sharing an edge



. Consequently, QUEEN typically identifies a higher number of neighbors for each region, as reflected in the provided summary results: the QUEEN method shows 490 nonzero links with an average of 4.9 links per region, while ROOK shows 462 nonzero links with an average of 4.62 links per region. This distinction influences the structure of the spatial weights matrix and can significantly impact spatial analyses, such as autocorrelation or regression, by altering the perceived spatial relationships and connectivity among regions.

**#Distance based (k nearest) neighbor** Top of Form

# Get coordinates from spatial data  
coords <- st\_coordinates(sids\_SP)   
  
# Assuming the first two columns are latitude and longitude  
coords\_2d <- coords[, 1:2]  
  
# Then, re-run the knn2nb process with the adjusted 2D coordinates  
sids\_kn1 <- knn2nb(knearneigh(coords\_2d, k=1))  
sids\_kn2 <- knn2nb(knearneigh(coords\_2d, k=2))  
sids\_kn4 <- knn2nb(knearneigh(coords\_2d, k=4))  
  
# Plot  
plot(sids\_SP, main = "SIDS Spatial Points")  
plot(sids\_kn1, coords, add = TRUE, col = "red")  
plot(sids\_kn2, coords, add = TRUE, col = "blue")  
plot(sids\_kn4, coords, add = TRUE, col = "green")

##### Result

## Warning in knearneigh(coords\_2d, k = 1): knearneigh: identical points found

## Warning in knearneigh(coords\_2d, k = 1): knearneigh: kd\_tree not available for  
## identical points

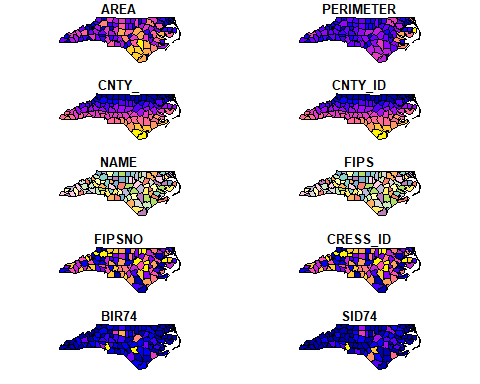
## Warning in knearneigh(coords\_2d, k = 2): knearneigh: identical points found

## Warning in knearneigh(coords\_2d, k = 2): knearneigh: kd\_tree not available for  
## identical points

## Warning in knearneigh(coords\_2d, k = 4): knearneigh: identical points found

## Warning in knearneigh(coords\_2d, k = 4): knearneigh: kd\_tree not available for  
## identical points

## Warning: plotting the first 10 out of 18 attributes; use max.plot = 18 to plot  
## all



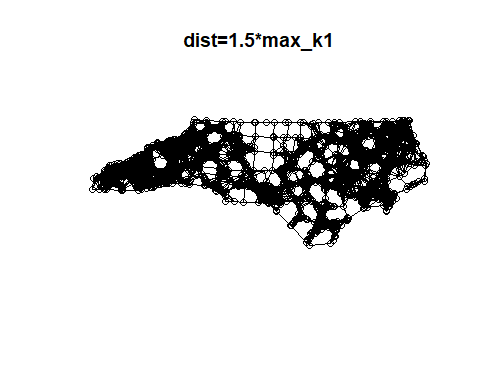
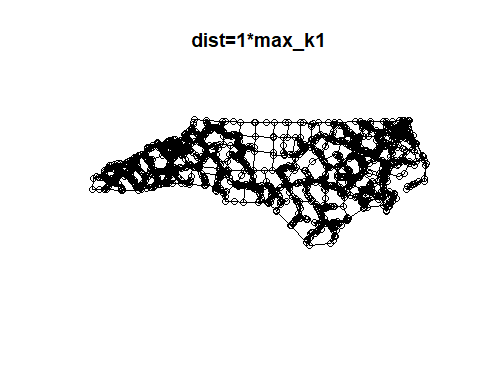
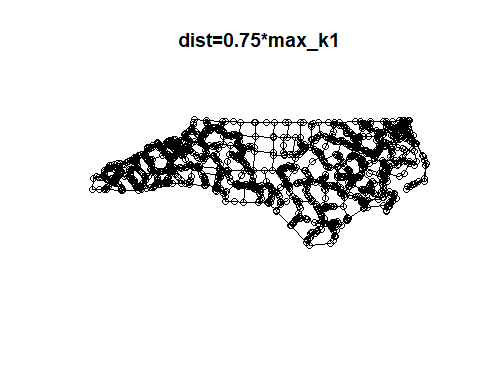
**Q4.**   
In the context of the provided maps, a K-Nearest Neighbors (KNN) analysis could be used to discern spatial patterns by identifying regions that are most similar to each other based on a selected attribute. For instance, a KNN approach applied to the "BIR74" or "SID74" maps would determine which regions have comparable birth rates or SIDS incidences, respectively, by finding the 'k' closest regions in terms of these specific attributes. Similarly, for the "AREA" or "PERIMETER" maps, KNN could identify regions with similar physical sizes or boundary lengths. While KNN is less applicable to categorical maps like "CNTY\_ID" or "FIPS," it could still be used if these identifiers are proxies for location-based attributes. Essentially, KNN leverages the spatial context of the data, allowing for the analysis of regional similarities and dissimilarities, which could be pivotal for targeted interventions or further geographical study.

### Assign neighbors based on a specified distance.

# Calculate distances  
dist <- unlist(nbdists(sids\_kn1, coords\_2d))  
summary(dist) # Summarize distances to understand distribution  
max\_k1 <- max(dist) # Find the maximum distance  
  
# Create distance-based neighborhoods  
sids\_kd1 <- dnearneigh(coords\_2d, d1=0, d2=0.75\*max\_k1)  
sids\_kd2 <- dnearneigh(coords\_2d, d1=0, d2=1\*max\_k1)  
sids\_kd3 <- dnearneigh(coords\_2d, d1=0, d2=1.5\*max\_k1)  
  
# Set up the plotting area to have 3 rows for the different distance thresholds  
par(mfrow=c(3,1))  
  
# Plot for dist=0.75\*max\_k1  
plot(st\_geometry(sids\_SP), main="dist=0.75\*max\_k1") # Plot the spatial object  
plot(sids\_kd1, coords, add=TRUE) # Add the dnearneigh plot  
  
# Plot for dist=1\*max\_k1  
plot(st\_geometry(sids\_SP), main="dist=1\*max\_k1") # Plot the spatial object again for a new comparison  
plot(sids\_kd2, coords, add=TRUE) # Add the dnearneigh plot  
  
# Plot for dist=1.5\*max\_k1  
plot(st\_geometry(sids\_SP), main="dist=1.5\*max\_k1") # Plot the spatial object again for another new comparison  
plot(sids\_kd3, coords, add=TRUE) # Add the dnearneigh plot

##### Result

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0 0.0 0.0 445.1 0.0 21614.7



### Q5. briefly describe their differences

The differences between the maps are in the density and connectivity of the neighbors. In the first map with the smallest distance threshold, spatial units are connected to fewer neighbors, resulting in a less dense network with more isolated clusters. As the distance threshold increases, as seen in the second map, more units are considered neighbors, leading to a denser network with more connections among the units. The third map shows the highest connectivity, with the largest distance threshold allowing units to reach even more neighbors, creating the densest network of the three. These increasing thresholds reflect a broader definition of neighborhood, with spatial units becoming neighbors over greater distances, which could be significant in analyses that depend on spatial relationships, such as clustering or spread of phenomena.

### Q6. Define neighbors based on a specified distance (the mean distance of neighbors). Take a screen shot of your plot and briefly describe their differences

The image depicts three maps of a geographic area, each applying a different distance threshold to define neighbors among spatial units. The first map uses a distance threshold of **0.75 \* max\_k1**, the second map uses **1 \* max\_k1**, and the third map **1.5 \* max\_k1**, where **max\_k1** likely represents the maximum distance from a reference point such as the farthest neighbor or some other specified distance metric in the dataset.

### Row-standardized weights matrix.

#Row-standardized weights matrix  
sids\_nbq\_w<- nb2listw(sids\_nbq)  
print(sids\_nbq\_w)  
sids\_nbq\_w$neighbours[1:5]

##### Result

## Characteristics of weights list object:  
## Neighbour list object:  
## Number of regions: 100   
## Number of nonzero links: 490   
## Percentage nonzero weights: 4.9   
## Average number of links: 4.9   
##   
## Weights style: W   
## Weights constants summary:  
## n nn S0 S1 S2  
## W 100 10000 100 44.65023 410.4746

## [[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

**Q7. What is the result? What does this mean?**

The result is a summary of a row-standardized spatial weights matrix for a set of 100 regions, indicating that each region has an average of 4.9 links to its neighbors. The row-standardization ("W" style) implies that the influence of each region's neighbors is normalized, making the analysis invariant to the number of neighbors. Essentially, this means that in subsequent spatial analyses, all regions will have the same total weight for their neighbors, ensuring that regions with many neighbors do not have a disproportionately high influence. The constants S0, S1, and S2 are summary statistics that will be used in global spatial autocorrelation calculations, which assess how much a region is similar to its neighbors across the entire study area. The individual lists for the first five regions provide a glimpse into the specific neighborhood structure, showing which regions are considered neighbors. For instance, region 1 is connected to regions 2, 18, and 19, indicating a direct relationship or proximity.

### Sids\_nbq weights matrixs

#Row-standardized weights matrix  
sids\_nbq\_w$weights[1:5]

##### Result

## [[1]]  
## [1] 0.3333333 0.3333333 0.3333333  
##   
## [[2]]  
## [1] 0.3333333 0.3333333 0.3333333  
##   
## [[3]]  
## [1] 0.2 0.2 0.2 0.2 0.2  
##   
## [[4]]  
## [1] 0.5 0.5  
##   
## [[5]]  
## [1] 0.25 0.25 0.25 0.25

**Q8. What is the result? What does this mean?**

The result indicates the row-standardized weights for the first five regions in the **sids\_nbq\_w** spatial weights matrix. Row-standardization means that the weights for each region's neighbors sum to 1, ensuring that the influence of neighboring regions is not biased by the number of neighbors a region has.

For region 1, the weights are evenly distributed among its three neighbors, each neighbor contributing a third (approximately 0.333) to the total weight. The same is true for region 2, which also has three neighbors with equal influence. Region 3, however, has five neighbors, each with a weight of 0.2, thereby summing to 1. Region 4 has two neighbors with equal weights of 0.5, and region 5 has four neighbors, each assigned a weight of 0.25.

This means that in spatial analyses that use this weights matrix, such as calculating spatial lag or conducting spatial regression, each region's influence on a focal region is adjusted to account for the number of neighbors.

### Binary Weights

sids\_nbq\_wb<-nb2listw(sids\_nbq, style="B")  
print(sids\_nbq\_wb)  
sids\_nbq\_wb$weights[1:5]

##### Result

## Characteristics of weights list object:  
## Neighbour list object:  
## Number of regions: 100   
## Number of nonzero links: 490   
## Percentage nonzero weights: 4.9   
## Average number of links: 4.9   
##   
## Weights style: B   
## Weights constants summary:  
## n nn S0 S1 S2  
## B 100 10000 490 980 10696

## [[1]]  
## [1] 1 1 1  
##   
## [[2]]  
## [1] 1 1 1  
##   
## [[3]]  
## [1] 1 1 1 1 1  
##   
## [[4]]  
## [1] 1 1  
##   
## [[5]]  
## [1] 1 1 1 1

**Q9. What is the result? What does this mean?**  
The result presents a binary spatial weights matrix (**sids\_nbq\_wb**) for 100 regions where each of the 490 nonzero links is assigned a weight of 1. The binary style ("B") indicates that all neighboring connections are considered equal regardless of their proximity or other characteristics; a neighbor is simply marked as "1" for present or "0" for absent, leading to a straightforward, unweighted adjacency matrix.

For example, the first five entries in the weights list show that regions 1, 2, and 4 are each connected to three neighbors, region 3 to five neighbors, and region 5 to four neighbors, all with a weight of 1. The summary statistics such as **S0** (sum of all weights), **S1** (twice the sum of all weights because each link is counted twice), and **S2** (sum of the squared weights) are used in spatial analysis calculations.

These binary weights matrix is typically used in spatial analyses that require a simple measure of connectivity without the need for nuanced differentiation between the strengths of spatial relationships.

### Inverse distance weighting bases weights on the distance between centroids

# Extract coordinates from the 'sf' object  
  
coords\_2d <- st\_coordinates(sids\_SP)[,1:2]  
  
sids\_nbq <- knn2nb(knearneigh(coords\_2d, k=2))   
dist <- nbdists(sids\_nbq, coords\_2d)  
  
# Adjusting IDW calculation to handle distances of 0  
idw <- lapply(dist, function(x) ifelse(x == 0, Inf, 1/(x/1000)))  
  
# Creating a spatial weights object with the IDW values  
sids\_nbq\_idwb <- nb2listw(sids\_nbq, glist=idw, style="B")  
  
# Accessing initial weights and providing summary  
initial\_weights <- sids\_nbq\_idwb$weights[1:5] # Get the first 5 weights  
print(initial\_weights)  
  
# Summary of all weights  
weights\_summary <- summary(unlist(sids\_nbq\_idwb$weights))  
print(weights\_summary)

##### Result

## Warning in knearneigh(coords\_2d, k = 2): knearneigh: identical points found

## Warning in knearneigh(coords\_2d, k = 2): knearneigh: kd\_tree not available for  
## identical points

## [[1]]  
## [1] Inf Inf  
##   
## [[2]]  
## [1] 0.525362 Inf  
##   
## [[3]]  
## [1] 0.525362 Inf  
##   
## [[4]]  
## [1] 0.1448819 Inf  
##   
## [[5]]  
## [1] 0.3010594 Inf

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.03513 0.35256 Inf Inf Inf Inf

**Q10. What is the result? What does this mean?**

The result indicates an attempt to create a spatial weights matrix using inverse distance weighting (IDW) based on the nearest neighbors (k=2) for a set of spatial data points. The warnings about identical points and the unavailability of a kd\_tree suggest that there are duplicate points in the dataset, which causes issues with the k-nearest neighbor calculation.

In the IDW calculation, distances of zero result in an infinite weight (**Inf**), as seen in the initial weights output. For the first region, both of its nearest neighbors are at a distance of zero, leading to infinite weights. The second and third regions have one neighbor with a zero distance and another with a small but nonzero distance, resulting in one infinite weight and one finite weight. Regions 4 and 5 show similar patterns with one infinite and one smaller finite weight.

The summary of all weights shows that the minimum, first quartile, and median are all finite values, while the mean, third quartile, and maximum are infinite. This suggests that while there are many regions with finite weights indicating varying degrees of closeness, the presence of infinite weighs heavily skews the data, reflecting the influence of those duplicate or extremely close points.

### Sids\_nbq weights matrixs

# Summary of all weights  
weights\_summary <- summary(unlist(sids\_nbq\_idwb$weights))  
print(weights\_summary)  
sids\_nbq\_w <- nb2listw(sids\_nbq, zero.policy = T)  
print(sids\_nbq\_w)

##### Result

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.03513 0.35256 Inf Inf Inf Inf

## Characteristics of weights list object:  
## Neighbour list object:  
## Number of regions: 2529   
## Number of nonzero links: 5058   
## Percentage nonzero weights: 0.07908264   
## Average number of links: 2   
## 396 disjoint connected subgraphs  
## Non-symmetric neighbours list  
##   
## Weights style: W   
## Weights constants summary:  
## n nn S0 S1 S2  
## W 2529 6395841 2529 2125.5 11416.5