Areal data

Lecture 8

Joaquin Cavieres

1. Areal data

1.1. Spatial neighborhood

The concept of spatial neighborhood is useful for the exploration of areal data to assess spatial autocorrelation and find out whether close areas have similar or dissimilar values.

Spatial neighbors can be defined in several ways depending on the variable of interest and the specific setting. The simplest neighborhood definition assumes that neighbors are areas that share a common border, perhaps a vertex. We can also expand the idea of neighborhood to include areas that are close, but not necessarily adjacent, by assuming neighbors are areas that are within some distance apart.

1.1.1. Neighbors based on contiguity

Using the function `poly2nb()` of the "spdep" package we can construct a list of neighbors based on areas with contiguous boundaries, that is, areas sharing one or more boundary point.

Specifically, `poly2nb()` accepts a list of polygons and returns a list of class `nb` with the neighbors of each area. The default type in poly2nb() is `queen = TRUE` so neighbors of a given area are other areas sharing a common point or more than one point.

```
# Load the libraries
library(spData)
library(sf)
library(spdep)
library(ggplot2)
map <- st_read(system.file("shapes/columbus.shp",
               package = "spData"), quiet = TRUE)


library(spdep)
nb <- spdep::poly2nb(map, queen = TRUE)
head(nb)
```

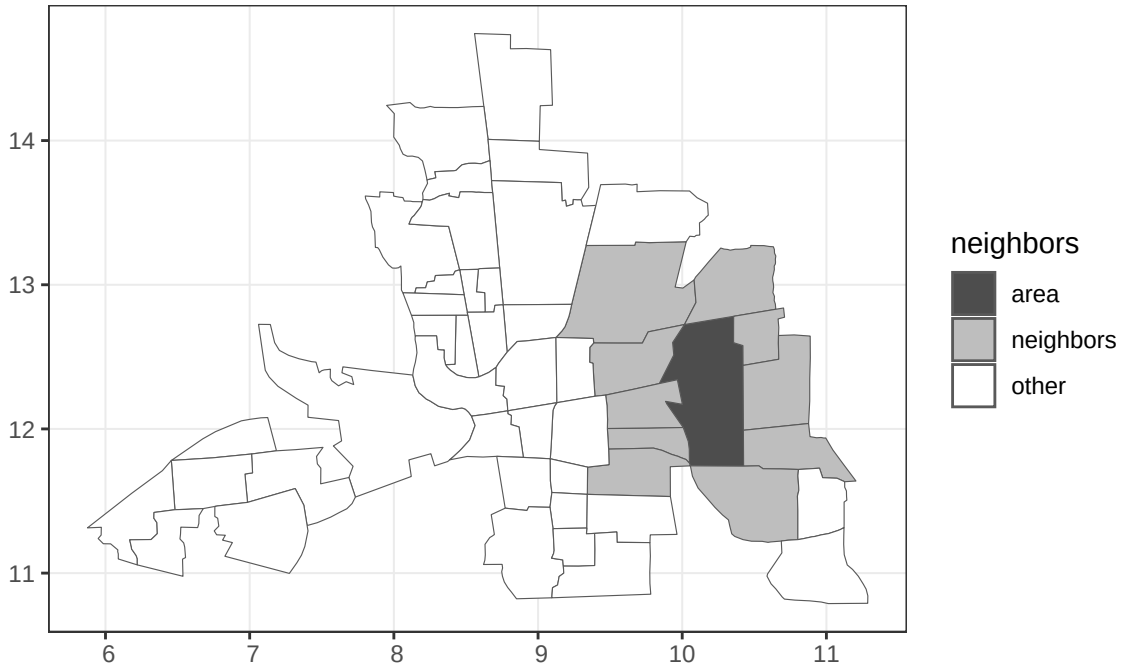
```
## [[1]]
## [1] 2 3
##
## [[2]]
## [1] 1 3 4
##
## [[3]]
## [1] 1 2 4 5
##
## [[4]]
## [1] 2 3 5 8
##
## [[5]]
## [1]  3  4  6  8  9 11 15 16
##
## [[6]]
## [1] 5 9
```

The next figure shows a map with the neighbors obtained. This plot is obtained by first plotting the map, and then overlapping the neighborhood structure with the `plot.nb()` function passing the neighbor list and the coordinates of the map.

```
plot(st_geometry(map), border = "lightgrey")
plot.nb(nb, st_geometry(map), add = TRUE)
```

We also can plot the the neighbors of a given area by adding a new column in `map` representing the neighbors of the area (here the area 20).

```
id <- 20 # area id
map$neighbors <- "other"
map$neighbors[id] <- "area"
map$neighbors[nb[[id]]] <- "neighbors"
ggplot(map) + geom_sf(aes(fill = neighbors)) + theme_bw() +
scale_fill_manual(values = c("gray30", "gray", "white"))
```



Given a neighbor list, the function `spdep::card()` give us the number of neighbors of each area. We can also obtain the number of neighbors of each area with `lengths(nb)`

```
spdep::card(nb)
```

```
##  [1]  2  3  4  4  8  2  4  6  8  4  5  6  4  6  6  8  3  4  3 10  3  6  3  7  8
## [26]  6  4  9  7  5  3  4  4  4  7  5  6  6  3  5  3  2  6  5  4  2  2  4  3
```

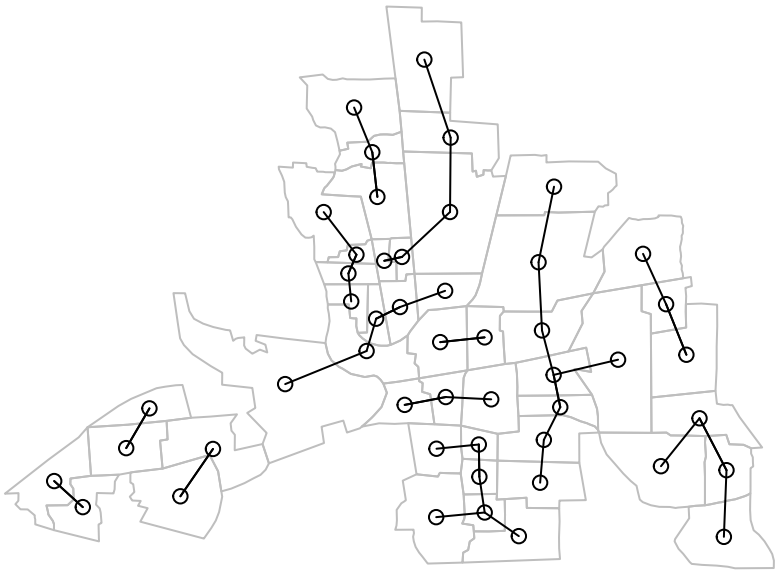
```
# Build a table with the areas' ids in rows, and the number of neighbors in columns
#table(1:nrow(map), card(nb))
```

1.1.2. Neighbors based on k nearest neighbors

We can consider as neighbors of an area its ik nearest neighbors based on the distance separating them as well. If we want to represent 3 nearest neighbors of an area, so using the function `knearneigh()` of the "spdep" package allows us to obtain a matrix with the indices points belonging to the set of the k nearest neighbors of each area.

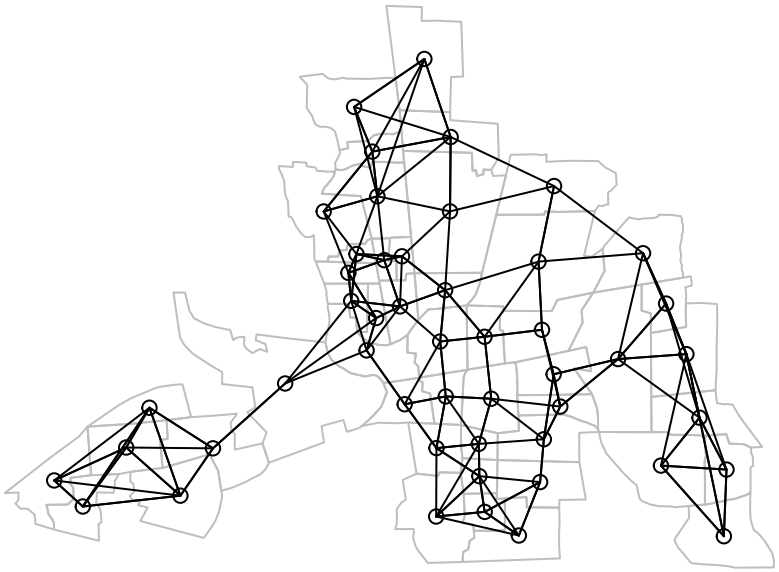
To use this function we need to declare the arguments as: a matrix of point coordinates and the number k of nearest neighbors. Firts, look in this example where you think that there are only 1 neighbor ($k = 1$)

```
columbus <- st_read(system.file("shapes/columbus.shp", package="spData")[1], quiet=TRUE)
locs <- st_centroid(st_geometry(columbus), of_largest_polygon=TRUE)
col.knn <- knearneigh(locs, k=1)
plot(st_geometry(columbus), border="grey")
plot(knn2nb(col.knn), locs, add=TRUE)
title(main="K nearest neighbours, k = 1")
```



Also, we can compare the previous map with a k nearest neighbours = 4,

```
columbus <- st_read(system.file("shapes/columbus.shp", package="spData")[1], quiet=TRUE)
locs <- st_centroid(st_geometry(columbus), of_largest_polygon=TRUE)
col.knn <- knearneigh(locs, k=4)
plot(st_geometry(columbus), border="grey")
plot(knn2nb(col.knn), locs, add=TRUE)
title(main="K nearest neighbours, k = 4")
```

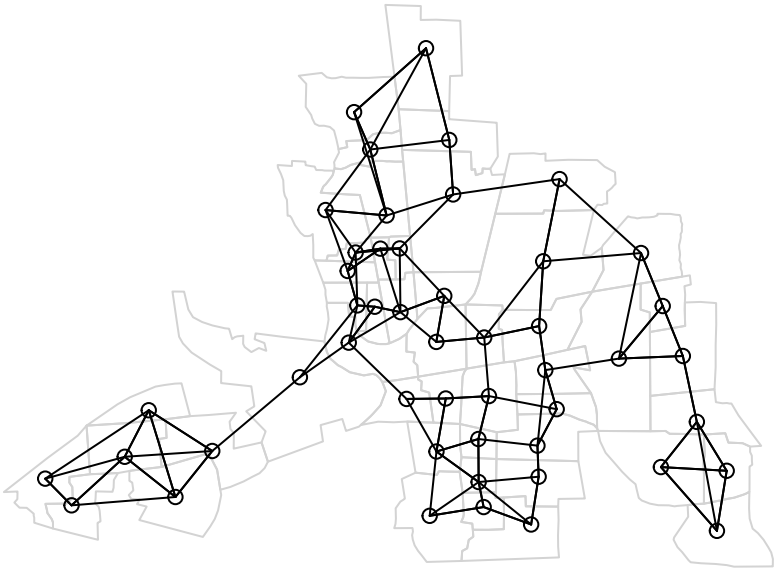


Finally, we will create a neighbors based on $k = 3$

```
# Neighbors based on 3 nearest neighbors
loc <- st_centroid(map) # mat of coordinates
head(loc, 5)
```

```
## Simple feature collection with 5 features and 21 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: 8.332658 ymin: 13.29637 xmax: 9.012265 ymax: 14.36908
## CRS:           NA
##       AREA PERIMETER COLUMBUS_ COLUMBUS_I POLYID NEIG  HOVAL    INC    CRIME
## 1 0.309441  2.440629         2          5      1    5 80.467 19.531 15.72598
## 2 0.259329  2.236939         3          1      2    1 44.567 21.232 18.80175
## 3 0.192468  2.187547         4          6      3    6 26.350 15.956 30.62678
## 4 0.083841  1.427635         5          2      4    2 33.200  4.477 32.38776
## 5 0.488888  2.997133         6          7      5    7 23.225 11.252 50.73151
##       OPEN    PLUMB DISCBD     X     Y NSA NSB EW CP THOUS NEIGNO
## 1 2.850747 0.217155   5.03 38.80 44.07   1   1  1  0  1000   1005
## 2 5.296720 0.320581   4.27 35.62 42.38   1   1  0  0  1000   1001
## 3 4.534649 0.374404   3.89 39.82 41.18   1   1  1  0  1000   1006
## 4 0.394427 1.186944   3.70 36.50 40.52   1   1  0  0  1000   1002
## 5 0.405664 0.624596   2.83 40.01 38.00   1   1  1  0  1000   1007
##                     geometry neighbors
## 1 POINT (8.827218 14.36908)     other
## 2 POINT (8.332658 14.03162)     other
## 3 POINT (9.012265 13.81972)     other
## 4 POINT (8.460801 13.71696)     other
## 5 POINT (9.007982 13.29637)     other
```

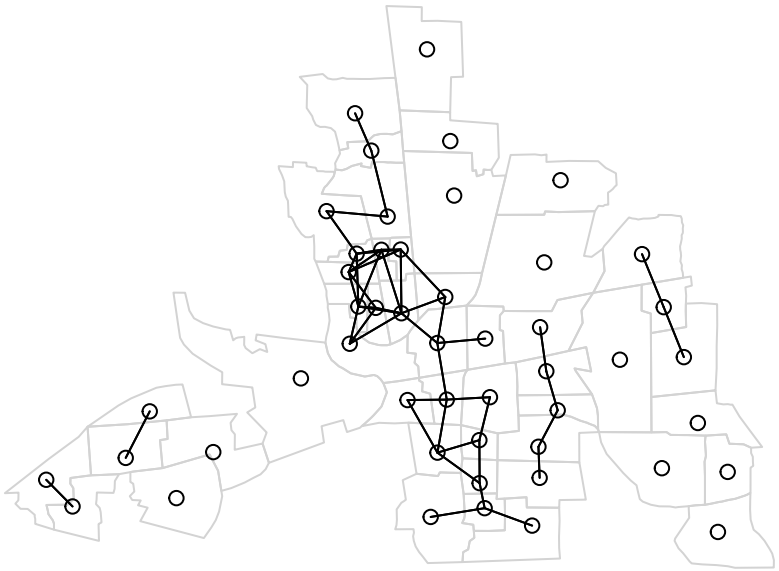
```
nb <- knn2nb(knearneigh(loc, k = 3)) # k number nearest neighbors
plot(st_geometry(map), border = "lightgrey")
plot.nb(nb, st_geometry(map), add = TRUE)
```



1.1.3. Neighbors based on distance

The arguments in the `dnearneigh()` function include the point coordinates (matrix), and the lower and upper distance bounds ($d1$ and $d2$). Here we can see the map of neighbors obtained by considering neighbors areas separated by a distance less than 0.4.

```
# Neighbors based on distance
nb <- dnearneigh(x = st_centroid(map), d1 = 0, d2 = 0.4)
plot(st_geometry(map), border = "lightgrey")
plot.nb(nb, st_geometry(map), add = TRUE)
```



Also we can propose an appropriate upper distance to ensure that eacgh area has at least k neighbors. In this case, we can use the `spdep::knearneigh()` passing the loc of coordinates (matrix) and the number of k chosen nearest neighbors to obtain a matrix with the k nearest neighbors of each area. After that, we have to use the `spdep::knn2nb()` to convert this list into a neighbor list of class `nb` with the ids of the neighbors.

```
loc <- st_centroid(map)
# k is the number nearest neighbors
nb1 <- knn2nb(knearneigh(loc, k = 1))
```

```
dist1 <- nbdists(nb1, loc)
summary(unlist(dist1))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1276  0.2543  0.3164  0.3291  0.4044  0.6189
```

In this example, the maximum distance is 0.6189 and we can take this value as an upper bound of the distance to ensure each area has at least one neighbor.

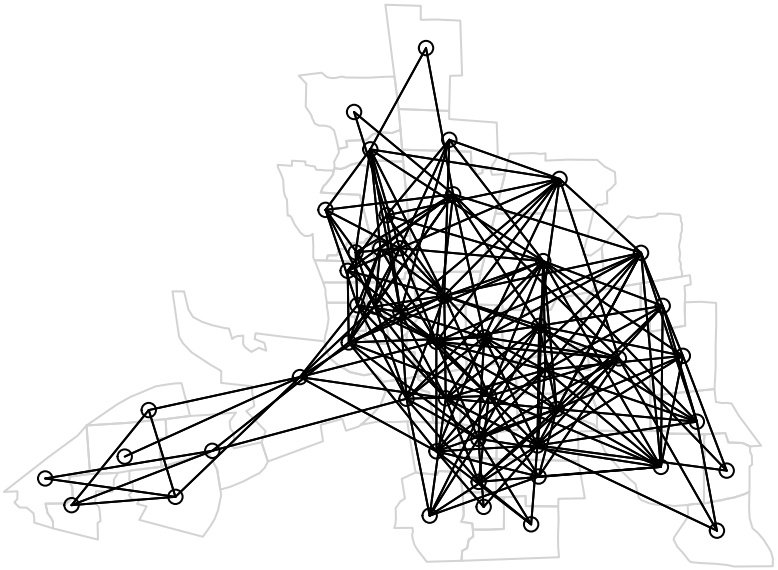
1.1.4. Neighbors of order k based on contiguity

By the `nblab()` function of the "spdep" package we can create higher order neighbor lists, where higher order neighbors are lags links from each other on the graph described by the input neighbors list of class `nb`. For this class, we need a list of neighbors of class `nb`, and the maximum lag considered. The result (an object) is a list of lagged neighbors lists each with class `nb`.

```
# library(spdep)
nb <- poly2nb(map, queen = TRUE)
nblags <- spdep::nblag(neighbours = nb, maxlag = 2)
```

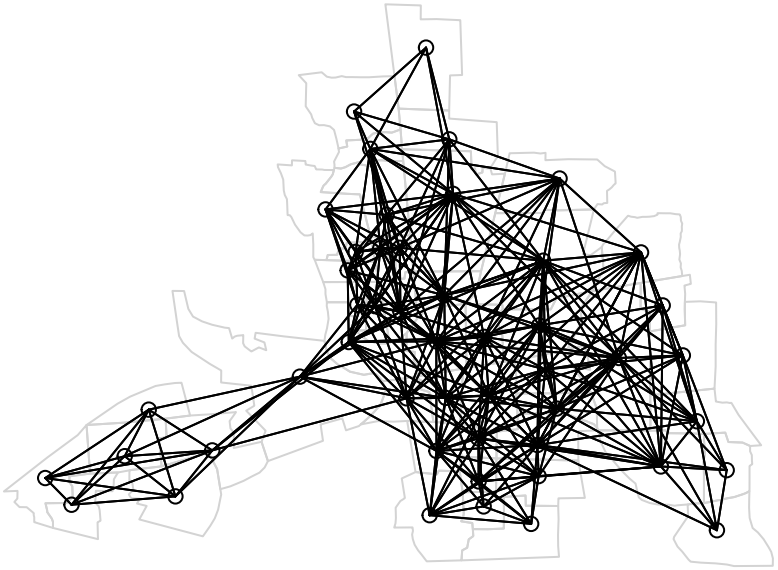
```r
# Neighbors of first order
plot(st_geometry(map), border = "lightgrey")
plot.nb(nblags[[1]], st_geometry(map), add = TRUE)
```

```r
# Neighbors of second order
plot(st_geometry(map), border = "lightgrey")
plot.nb(nblags[[2]], st_geometry(map), add = TRUE)
```



The function `nblag_cumul()` cumulates a list of lagged neighbors as the output of `nblag()` and returns a single neighbor list of class `nb` containing the neighbors of order 1 until the maximum lag considered.

```
# Neighbors of order 1 until order 2
nb <- spdep::poly2nb(map, queen = TRUE)
nblagsc <- spdep::nblag_cumul(nblags)
plot(st_geometry(map), border = "lightgrey")
plot.nb(nblagsc, st_geometry(map), add = TRUE)
```



1.1.4. Neighborhood matrices

We will create a spatial weights matrix based on a binary neighbor list, for this the function `nb2listw()` of the "spdep" package can be used to construct a spatial neighborhood matrix containing the spatial weights.

The neighbors can be binary or based on inverse distance values. To compute a spatial weights matrix based on a binary neighbor list, we use the `nb2listw()` function with the following arguments:

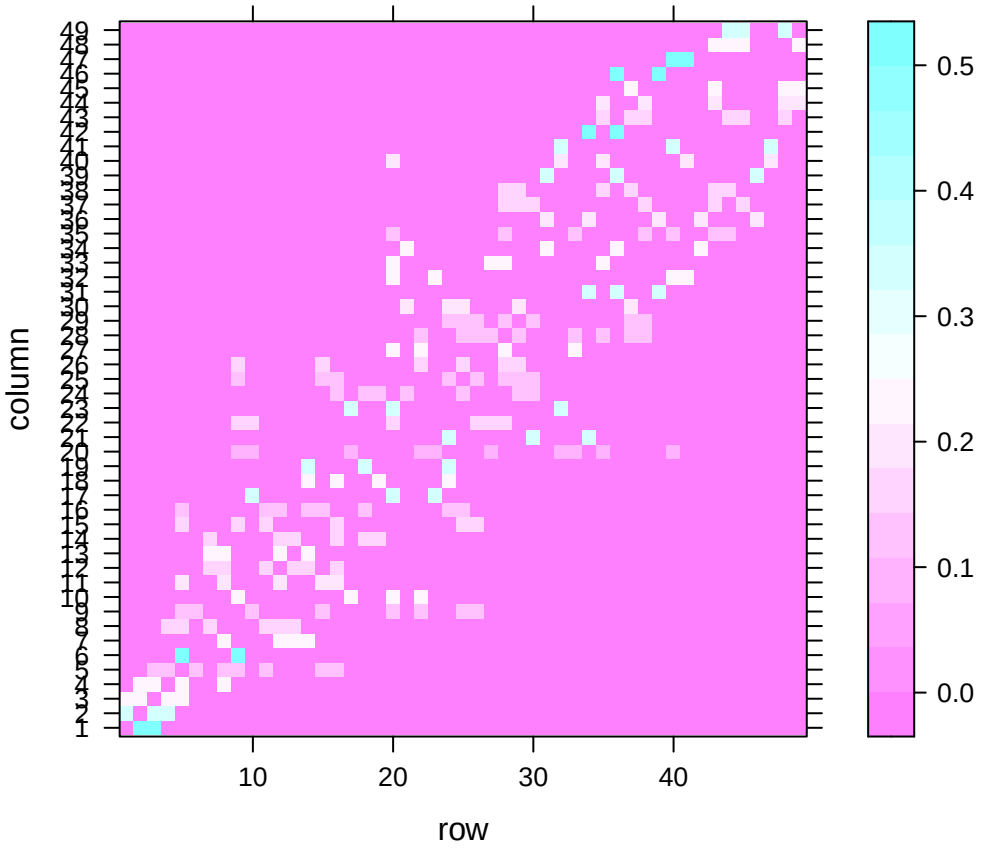
- `nb` list with neighbors,
- `style` indicates the coding scheme chosen. For example, `style = B` is the basic binary coding, and `W` is row standardized (1/number of neighbors),
- `zero.policy` is used to take into account regions with 0 neighbors. Specifically, `zero.policy = TRUE` permits the weight list to contain zero-length weights vectors, and `zero.policy = FALSE` stops the function with an error if there are empty neighbor sets.

```
nb <- poly2nb(map, queen = TRUE)
nbw <- spdep::nb2listw(nb, style = "W")
head(nbw$weights)
```

```
## [[1]]
## [1] 0.5 0.5
##
## [[2]]
## [1] 0.3333333 0.3333333 0.3333333
##
## [[3]]
## [1] 0.25 0.25 0.25 0.25
##
## [[4]]
## [1] 0.25 0.25 0.25 0.25
##
## [[5]]
## [1] 0.125 0.125 0.125 0.125 0.125 0.125 0.125 0.125
##
## [[6]]
## [1] 0.5 0.5
```

Visualizing the spatial weight matrix (it is a simple matrix with the weights built by the function `listw2mat()`), and using the function `lattice::levelplot()`:

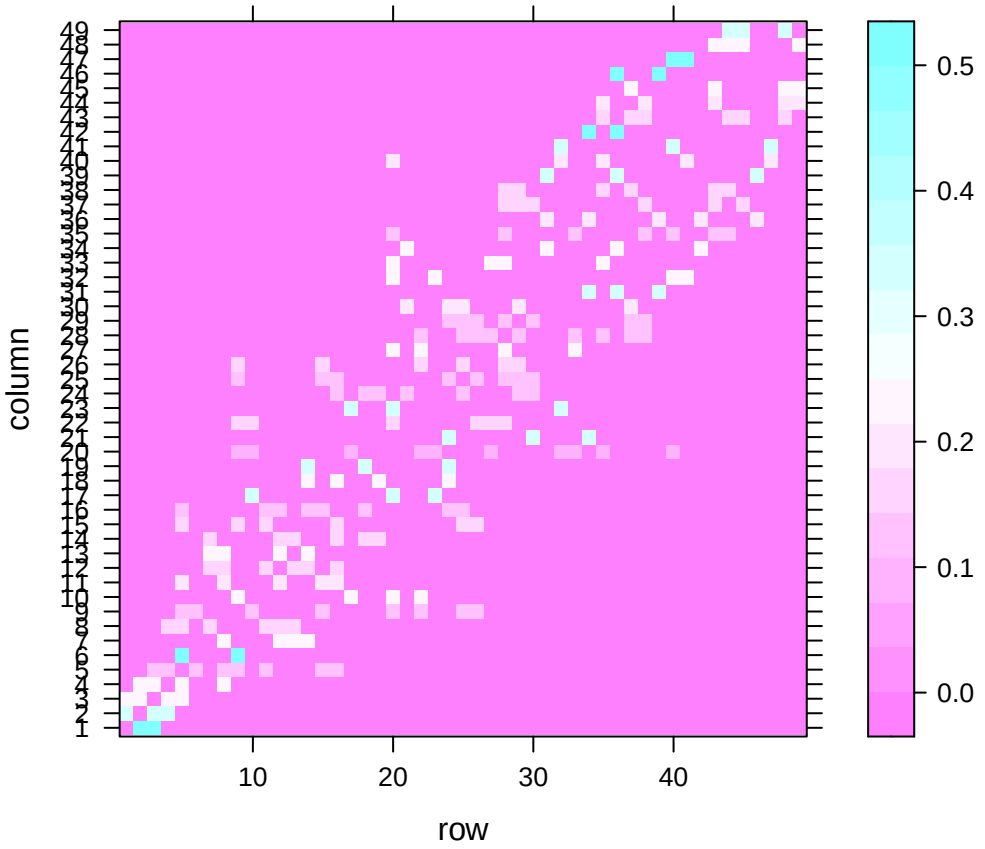
```
w_mat1 <- listw2mat(nbw)
lattice::levelplot(t(w_mat1))
```



Finally, to calculate the spatial weights matrix based on inverse distance values, we can use the `nbdists()` function to compute the distances along the links. Then, we can construct the list with spatial weights based on inverse distance values using `nb2listw()` where the argument `glist` is equal to a list of general weights corresponding to neighbors.

```
loc <- st_centroid(map)
nb <- poly2nb(map, queen = TRUE)
dists <- nbdists(nb, loc)
ids <- lapply(dists, function(x){1/x})

w_mat2 <- listw2mat(nbw)
lattice::levelplot(t(w_mat2))
```



References

- Moraga, P., 2023. Spatial Statistics for Data Science: Theory and Practice with R
- Bivand, R. S., Pebesma, E. J., Gomez-Rubio, V., & Pebesma, E. J. (2008). Applied spatial data analysis with R (Vol. 747248717, pp. 237-268). New York: Springer.
- Spatial Data Science with R and "terra". https://rspatial.org/index.html.