UNIVERSIDAD NACIONAL DE COLOMBIA

Modelado de datos en áreas

Estadística espacial

Daniela Arbeláez Montoya
Jefferson Gamboa Betancur
Jean Paul Piedrahita García

Universidad Nacional de Colombia
Ciencias, Escuela de Estadística
Medellín, Colombia
2021

Índice

1. Introducción

A lo largo del desarrollo de este documento, se mostrará la construcción de vecinos y los pesos que se pueden aplicar a los vecindarios. Una vez que este importante y a menudo exigente prerrequisito esté en su lugar, se procede a buscar formas de medir la autocorrelación espacial.

Si bien las pruebas se basan en modelos de procesos espaciales, primero se examinan las pruebas y solo posteriormente se pasa al modelado. También es interesante mostrar cómo se puede introducir la autocorrelación espacial en datos independientes, de modo que se puedan realizar simulaciones.

El conjunto de datos con el que se trabajará en esta ocasión, contiene 281 distritos censales para ocho condados centrales del estado de Nueva York complementado con límites de tramo. El área tiene una extensión de unos 160 km de norte a sur y de 120 km de este a oeste.

```
> library(rgdal)
> library(sf)
> library(spdep)
```

```
> NY8 <- readOGR("Base de datos", "NY8_utm18")
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "G:\Mi unidad\Universidad Nacional de Colombia\13. Treceavo semestre\Estadísti
## with 281 features
## It has 17 fields
```

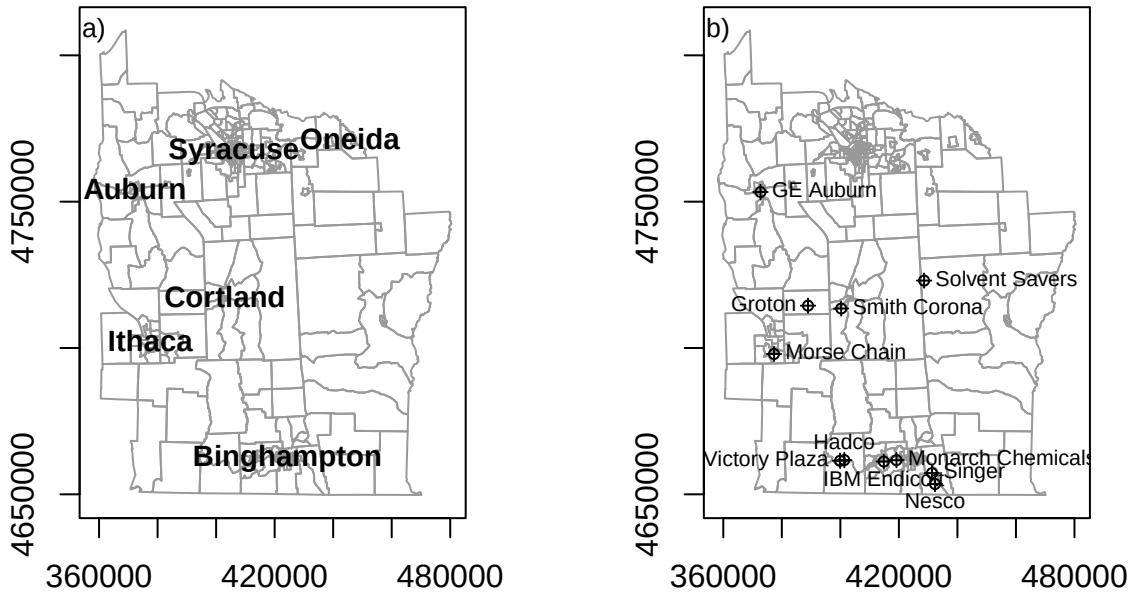
```
> TCE <- readOGR("Base de datos", "TCE")
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "G:\Mi unidad\Universidad Nacional de Colombia\13. Treceavo semestre\Estadísti
## with 11 features
## It has 5 fields
```

```
> cities <- readOGR("Base de datos", "NY8cities")
```

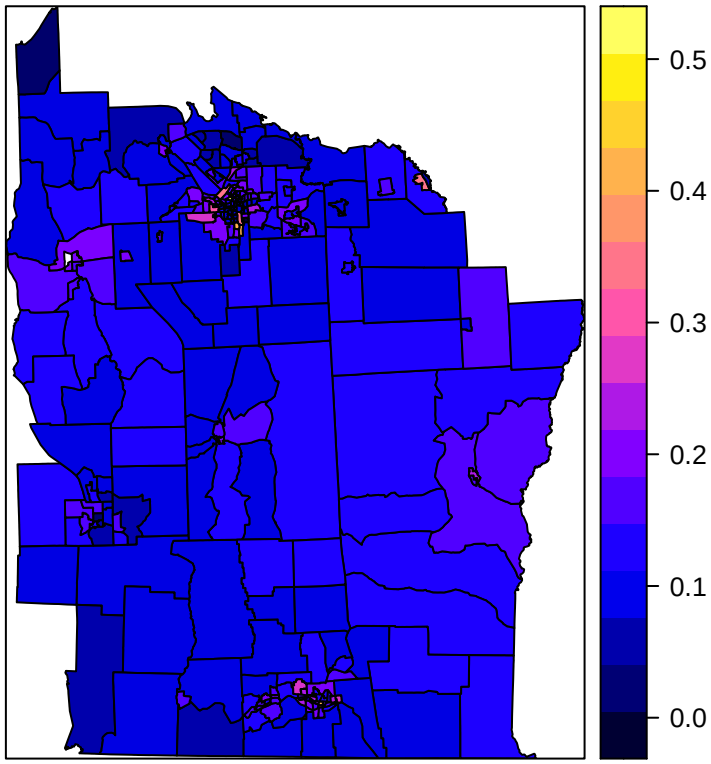
```
## OGR data source with driver: ESRI Shapefile
## Source: "G:\Mi unidad\Universidad Nacional de Colombia\13. Treceavo semestre\Estadísti
## with 6 features
## It has 1 fields
```

```
> par(mfrow=c(1,2))
> plot(NY8, border="grey60", axes=TRUE)
> text(coordinates(cities), labels=as.character(cities$names), font=2, cex=0.9)
> text(bbox(NY8)[1,1], bbox(NY8)[2,2], labels="a)", cex=0.8)
> plot(NY8, border="grey60", axes=TRUE)
> points(TCE, pch=1, cex=0.7)
> points(TCE, pch=3, cex=0.7)
> text(coordinates(TCE), labels=as.character(TCE$name), cex=0.7,
+  font=1, pos=c(4,1,4,1,4,4,4,2,3,4,2), offset=0.3)
> text(bbox(NY8)[1,1], bbox(NY8)[2,2], labels="b)", cex=0.8)
```

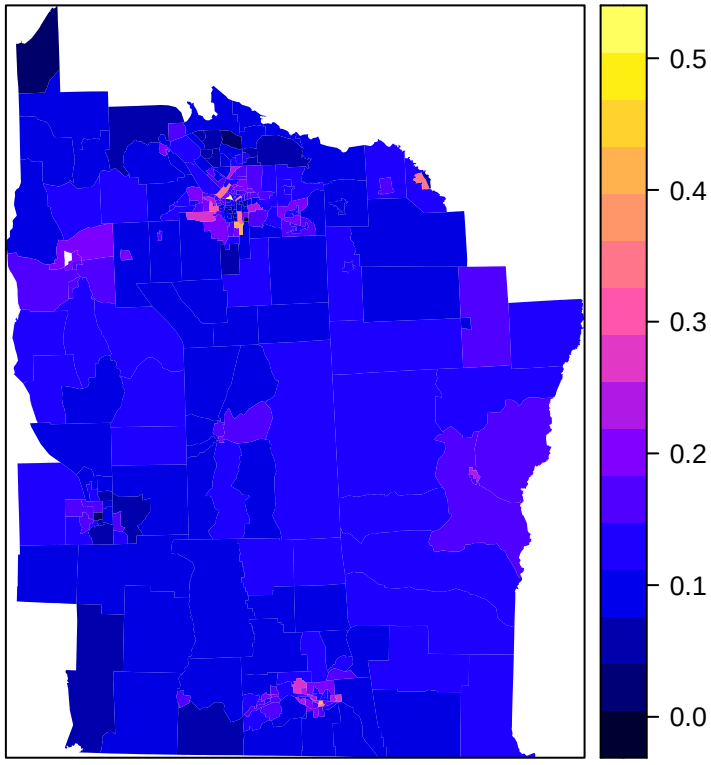


La figura *a*) muestra las principales ciudades en el área de estudio y *b*) la ubicación de 11 sitios de desechos peligrosos.

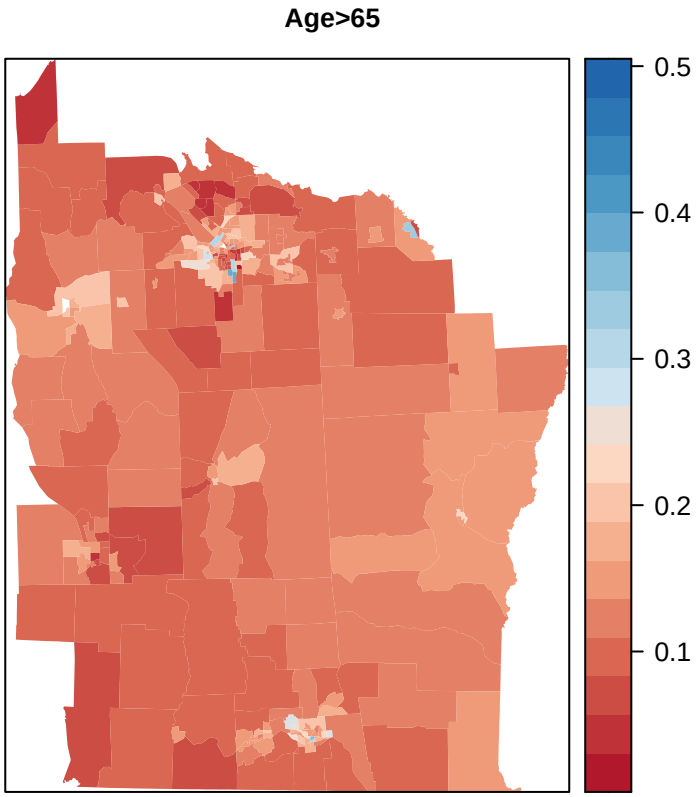
```
> spplot(NY8, c("PCTAGE65P"))
```




```
> spplot(NY8, c("PCTAGE65P"), col="transparent")
```



```
> library("RColorBrewer")
> #color palette creator function
> rds <- colorRampPalette(brewer.pal(8, "RdBu"))
> #get a range for the values
> tr_at <- seq(min(NY8$PCTAGE65P), max(NY8$PCTAGE65P), length.out=20)
> #create a color interpolating function taking the required
> #number of shades as argument
> tr_rds <- rds(20)
> #parameters
> # at - at which values colors change
> # col.regions - specify fill colors
> tr_pl <- spplot(NY8, c("PCTAGE65P"), at=tr_at, col="transparent",
+                 col.regions=tr_rds, main=list(label="Age>65", cex=0.8))
> plot(tr_pl)
```



2. Vecinos espaciales y peso espacial

La creación de ponderaciones espaciales es un paso necesario en el uso de datos de área, quizás solo para verificar que no haya patrones espaciales restantes en los residuos. El primer paso es definir a qué relaciones entre observaciones se les dará un peso distinto de cero, es decir, elegir el criterio de vecino que se usará; el segundo es asignar pesos a los enlaces vecinos identificados.

Tratar de detectar patrones en mapas de residuos visualmente no es una opción aceptable, por lo que se incluyen un montón de funciones en el paquete **spdep** para ayudar.

Las viñetas **"nb","CO69"** y **"sids"** en **spdep** incluyen discusiones sobre la creación y el uso de ponderaciones espaciales, y se pueden acceder a ellas de la siguiente manera:

```
> vignette("nb",package = "spdep")
```

2.1. Objetos vecinos

En el paquete spdep, las relaciones vecinas entre **n** observaciones están representadas por un objeto de clase **nb**. Es una lista de longitud n con los números de índice de vecinos de cada componente registrados como un vector entero. Si alguna observación no tiene vecinos, el componente contiene un número entero cero. También contiene atributos, típicamente un vector de identificadores de región de caracteres y un valor lógico que indica si las relaciones son simétricas. Los identificadores de región pueden usarse para verificar la integridad entre los datos mismos y el objeto vecino.

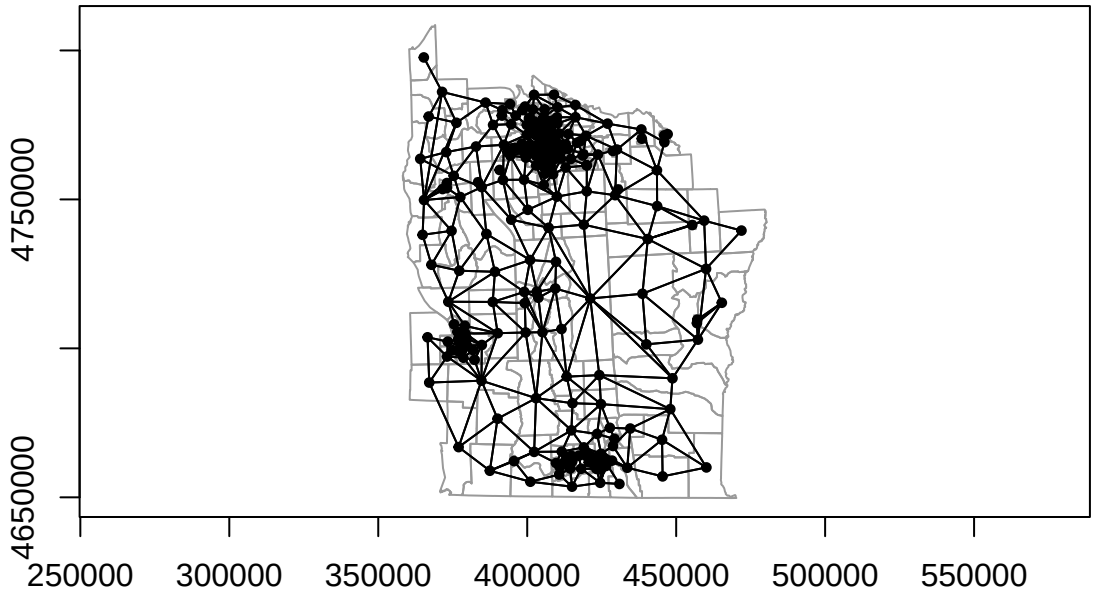
La función auxiliar **card** devuelve la cardinalidad del conjunto de vecinos para cada objeto, es decir, el número de vecinos.

```
> # reads a GAL lattice file into a neighbors list
> NY_nb <- read.gal("Base de datos/NY_nb.gal", region.id = row.names(NY8))
> summary(NY_nb)
```

```
## Neighbour list object:
## Number of regions: 281
## Number of nonzero links: 1522
## Percentage nonzero weights: 1.927534
## Average number of links: 5.41637
## Link number distribution:
##
```

```
##  1  2  3  4  5  6  7  8  9 10 11
##  6 11 28 45 59 49 45 23 10  3  2
## 6 least connected regions:
## 55 97 100 101 244 245 with 1 link
## 2 most connected regions:
## 34 82 with 11 links
```

```
> par(mfrow=c(1,1))
> plot(NY8, border="grey60", axes=TRUE)
> plot(NY_nb, coordinates(NY8), pch=19, cex=0.6, add=TRUE)
```



La figura muestra el gráfico vecino completo para el área de estudio de ocho condados.

Como ahora se tiene un objeto nb para examinar, se pueden presentar los métodos estándar para estos objetos. Hay métodos de impresión, resumen, diagrama y otros; el método de resumen presenta una tabla de la distribución del número de enlace, y tanto el método de impresión como el de resumen informan de la asimetría y la presencia de observaciones sin vecinos; la asimetría está presente cuando i es un vecino de j pero j no es un vecino de i.

Con motivos de simplicidad al mostrar como crear objetos vecinos, se trabaja con un subconjunto

del mapa que consta de los censos dentro de Syracuse, aunque los mismos principios se aplican al conjunto de datos completo.

```
> Syracuse <- NY8[NY8$AREANAME == "Syracuse city",]
> Sy0_nb <- subset(NY_nb, NY8$AREANAME == "Syracuse city")
> summary(Sy0_nb)
```

```
## Neighbour list object:
## Number of regions: 63
## Number of nonzero links: 346
## Percentage nonzero weights: 8.717561
## Average number of links: 5.492063
## Link number distribution:
##
##  1  2  3  4  5  6  7  8  9
##  1  1  5  9 14 17  9  6  1
## 1 least connected region:
## 164 with 1 link
## 1 most connected region:
## 136 with 9 links
```

```
> coords <- coordinates(Syracuse)
> IDs <- row.names(Syracuse)
> Sy8_nb <- knn2nb(knearneigh(coords, k = 1), row.names = IDs)
> Sy9_nb <- knn2nb(knearneigh(coords, k = 2), row.names = IDs)
> Sy10_nb <- knn2nb(knearneigh(coords, k = 4), row.names = IDs)
> dsts <- unlist(nbdists(Sy8_nb, coords))
> Sy11_nb <- dnearneigh(coords, d1 = 0, d2 = 0.75 * max(dsts),
+                       row.names = IDs)
```

2.2. Objetos de ponderaciones espaciales

La ponderación espacial se puede ver como una lista de ponderaciones espaciales indexadas por una lista de vecinos entre i y j esel *k-ésimo* elemento del *i-ésimo* componente de la lista de ponderaciones, y k nos dice cual de los *i-ésimos* valores del componente de la lista de vecinos es

igual a j. Si j no está presente en el i-*ésimo* componente de la lista de vecinos, j no es un vecino de i. Por ello algunas de las ponderaciones w_{ij} de la matriz de ponderaciones serán cero.

Una vez se establece la lista de los conjuntos de los vecinos en nuestra área de estudio, se procese a asignar los pesos espaciales, también es de tener en cuenta que se utiliza una notación bionaria cuando se sabe poco del proceso especial, dónde no hay una relación de vecino se pone 0 (cero) en caso contrario será la unidad.

La función `nb2listw` toma una lista de vecinos y lo convierte en un objeto de pesos. La conversión de los pesos se hace con un estilo W que se hace bajo una estandarización por fila para sumar la unidad. El método de impresión *(print)* para los objetos listw muestran las características de los vecinos subyacentes, el estilo de las ponderaciones espaciales y las constantes de las ponderaciones espaciales utilizadas en el cálculo de las pruebas de auto-correlación espacial. El componente de vecinos del objeto es el objeto nb subyacente, que proporciona la indexación del componente de ponderaciones.

```
> Sy0_lw_W <- nb2listw(Sy0_nb); Sy0_lw_W
```

```
## Characteristics of weights list object:
## Neighbour list object:
## Number of regions: 63
## Number of nonzero links: 346
## Percentage nonzero weights: 8.717561
## Average number of links: 5.492063
##
## Weights style: W
## Weights constants summary:
##    n   nn S0       S1      S2
## W 63 3969 63 24.78291 258.564
```

```
> names(Sy0_lw_W)
```

```
## [1] "style"      "neighbours" "weights"
```

```
> names(attributes(Sy0_lw_W))
```

```
## [1] "names"     "class"     "region.id" "call"      "GeoDa"
```

Para el `style = "W"`, los pesos varían entre la unidad divida por el mayor y el menor número de vecinos, y las sumas de los pesos para cada entidad de área son la unidad. Este estilo se puede interpretar como el **valor medio entre vecinos**. Los pesos de los enlaces que se originan en áreas con pocos vecinos son mayores que los que se originan en áreas con muchos vecinos, quizá aumentando las entidades de área en el borde del área de estudio sin querer. Esta representación ya no es simétrica, pero es similar tiende a ser simétrica.

```
> 1/rev(range(card(Sy0_lw_W$neighbours)))
```

```
## [1] 0.1111111 1.0000000
```

```
> summary(unlist(Sy0_lw_W$weights))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1111  0.1429  0.1667  0.1821  0.2000  1.0000
```

```
> summary(sapply(Sy0_lw_W$weights, sum))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1       1       1       1       1       1
```

Las funciones:

- `card` cuenta el número de vecinos en cada región en la lista de vecinos.
- `range` toma el valor mínimo y el máximo.
- `rev` revierte el vector
- `unlist` saca la información o datos atómicos de una lista
- `sapply` aplica la función suma a cada uno de los vectores asociado a las áreas.

La configuración de `style = "B"` - *Binario* retiene un peso de unidad para cada relación de vecino, pero en este caso, las sumas de pesos de las áreas difieren según el número de áreas vecinas que tienen.

```
> Sy0_lw_B <- nb2listw(Sy0_nb, style = "B")
> summary(unlist(Sy0_lw_B$weights))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1       1       1       1       1       1
```

```
> summary(sapply(Sy0_lw_B$weights, sum))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   4.500   6.000   5.492   6.500   9.000
```

El argumento `glist` se puede utilizar para pasar una lista de vectores de pesos generales correspondientes a las relaciones vecinas a `nb2list`. En este ejecicio se cree que la fuerza de las relaciones con los vecinos se atenúa con la distancia, por ello se estable en los pesos para que sean proporcionales ala distancia inversa entre los puntos que representan las áreas, usando `nbdists`para calcular las distancias para el objeto `nb` dado. Luego se usa `lapply` para invertir las distancias, obteniendo una estructura de pesos espaciales diferente a los anteriores. Si no se tiene ninguna razón para asumir más conocimiento sobre las relaciones con los vecinos que su existencia o ausencia, este paso es potencialmente engañoso. Si se sabe que el flujo de desplazamiento describen la estructura de las ponderaciones mejor que la alternativa binaria, puede valer la pena utilizarlas como ponderaciones generales; Sin embargo, puede haber problemas de simetría, porque tales flujos a diferencia de las distancias inversas, rara vez son simétricas.

```
> dsts <- nbdists(Sy0_nb, coordinates(Syracuse))
> idw <- lapply(dsts, function(x) 1/(x/1000))
> Sy0_lw_idwB <- nb2listw(Sy0_nb, glist = idw, style = "B")
> summary(unlist(Sy0_lw_idwB$weights))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.3886  0.7374  0.9259  0.9963  1.1910  2.5274
```

```
> summary(sapply(Sy0_lw_idwB$weights, sum))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.304   3.986   5.869   5.471   6.737   9.435
```

La función:

- `nbdists` devuelve las distancias euclidianas dadas por un vector de enlaces de vecinos (un objeto nb).
- `nb2listw` toma una lista de vecinos y lo convierte en un objeto de pesos.
- `lapply` aplica a cada distancia euclidiana el inverso.

La siguiente figura muestra tres representaciones de ponderaciones espaciales para Syracuse mostradas como matrices. La imagen `style = "W"` de la izquierda es asimétrica, con colores más oscuros ue muestran pesos más gandes para áreas con pocos vecinos. Los otros dos paneles son simétricos, pero expresan diferentes suposiciones sobre las fortalezas de las relaciones con los vecinos.

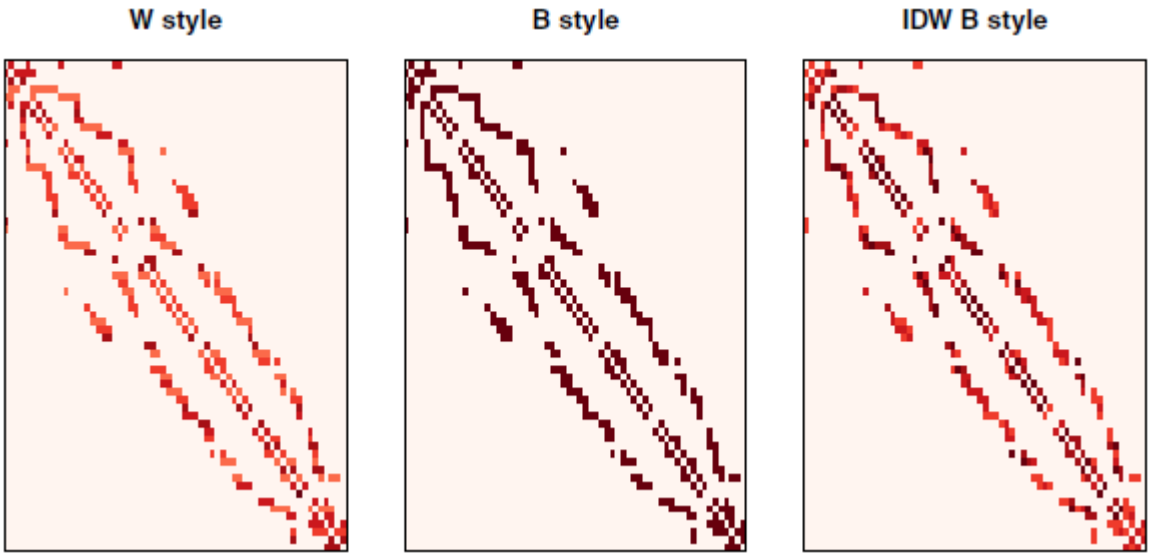


Figura 1: Tres representaciones de ponderaciones espaciales para Syracuse

La función `nb2listw` permite manejar listas de vecinos con áreas sin vecinos. Esto es debido porque la representación del conjunto vacío sea cero y debería ser representado como `NA` pero esto generaría problemas más adelante.

Es por esta razón que el argumento predeterminado es `cero.policy=FALSE`, lo que genera un error cuando se proporciona un argumento `nb` con áreas sin vecinos. Con el argumento `TRUE`permite la creación del objeto de ponderaciones espaciales, con ponderaciones cero.

```
> Sy0_lw_D1 <- nb2listw(Sy11_nb, style = "B")
```

```
## Error in nb2listw(Sy11_nb, style = "B"): Empty neighbour sets found
```

```
> Sy0_lw_D1 <- nb2listw(Sy11_nb, style = "B", zero.policy = TRUE)
> print(Sy0_lw_D1, zero.policy = TRUE)
```

```
## Characteristics of weights list object:
## Neighbour list object:
## Number of regions: 63
## Number of nonzero links: 230
```

```
## Percentage nonzero weights: 5.794911
## Average number of links: 3.650794
## 2 regions with no links:
## 154 168
##
## Weights style: B
## Weights constants summary:
##    n   nn  S0  S1   S2
## B 61 3721 230 460 4496
```

Un problema paralelo de los conjuntos de datos con valores perdidos en las variables pero con ponderaciones espaciales especificadas se aborda mediante el método `subset.listw`, que vuelve a generar las ponderaciones para el subconjunto de áreas dado, por ejemplo, dado por `complete.cases`. Sabiendo qué observaciones están incompletas, los vecinos subyacentes y las ponderaciones se pueden subdividir en algunos casos, con el objetivo de evitar la propagación de los valores `NA` al calcular los valores con retraso espacial. Muchas pruebas y funciones de ajuste de modelos pueden llevar a cabo esto internamente si se establece el indicador de argumento apropiado, aunque el analista cuidadoso preferirá subconjuntos de los datos de entrada y los pesos antes de probar o modelar.

2.3. Manejo de objetos de ponderaciones espaciales

Hay varios paquetes contribuidos que brindan soporte para matrices dispersas, entre las cuales `Matrix` es un paquete recomendado. La envoltura `as_dgRMatrix_listw` convierte un objeto `listw` en la matriz dispersa de formato orientado a filas comprimidas ordenadas por `Matrix`, como un objeto `dgRMatrix`, una subclase de la clase virtual `RsparseMatrix`. Es más fácil hacer una matriz dispersa orientada a filas a partir de un objeto de ponderaciones espaciales, ya que las ponderaciones están orientadas a filas. Una función que se usa mucho dentro de las funciones de prueba y ajuste de modelos es `listw2U`, que devuelve un objeto `listw` simétrico que representa la matriz de ponderaciones espaciales $\frac{1}{2}(W + W^T)$

Los objetos vecinos y de pesos pueden ser producidos en otros software e importarse a R y además se pueden exportar sin dificultad. Como ejemplo se han generado algunos archivos de GeoDa[1] a partir de distritos censales de Syracuse escritos como un shapefile, con el centroide utilizando

[1] https://sgsup.asu.edu/geodacenter-redirect

aquí almacenado en el marco de datos. Los dos primeros son para vecinos de contigüidad. Estos archivos denomidos en formato GAL contienen solo información de vecinos y se describen en detalle en el archivo de ayuda que acompaña a la función `read.gal`

```
> Sy14_nb <- read.gal("Base de datos/Sy_GeoDa1.GAL")
> isTRUE(all.equal(Sy0_nb, Sy14_nb, check.attributes = FALSE))
```

```
## [1] TRUE
```

```
> Sy16_nb <- read.gwt2nb("Base de datos/Sy_GeoDa4.GWT")
> isTRUE(all.equal(Sy10_nb, Sy16_nb, check.attributes = FALSE))
```

```
## [1] TRUE
```

2.4. Uso de pesos para simular la autocorrelación espacial

3. Prueba de autocorrelación espacial

3.1. Pruebas globales

3.2. Pruebas locales

4. Ajuste de modelos de datos de área

4.1. Enfoques de estadística espacial

4.1.1. Modelos autorregresivos simultáneos

4.1.2. Modelos autorregresivos condicionales

4.1.3. Ajuste de modelos de regresión espacial