M1683. Intensificación en Saberes Técnicos: Diseño y Gestión de Bases de Datos Territoriales. Regresión espacial I

Domingo Rasilla

30 diciembre, 2021

Contents

1. INTRODUCCIÓN

Inicialmente, revisaremos alguno de los diferentes métodos para construir una matriz espacial de ponderaciones. Posteriormente, pasaremos por el proceso de analizar y modelizar datos de tipo areal

Antes de comenzar la sesión, el alumno debará crear una nueva carpeta en el directorio de trabajo, que se llamará `sesion03`. Dicha carpeta deberá contener los siguientes ficheros:

- Datos de enfermos de leucemia en New York (*NY_data.zip*).
- Precios de la vivienda en Boston (*boston.tr.zip* - contiene los límites de las entidades espaciales de la ciudad).
- Tasas de criminalidad en Columbus, Ohio (*columbus.zip*).
- Precios e impuestos de coches usados (*usedcars.zip*).

Recuerda que estos ficheros están comprimidos, por lo que deberán ser descomprimidos para trabajar con ellos. Tras abrir RStudio, cambia el fichero de trabajo por defecto a `sesion03`.

Para realizar las próximas tareas, tendrás que cargar en RStudio una serie de paquetes. Algunos son necesarios para el análisis espacial (**sf**, **spdep** y **spatialreg**); otro servirá para crear mapas (**tmap**) mientras que **ggplot2** sirve para elaborar gráficos.

```
pkgs <- c("sf",
          "spdep",
          "spatialreg",
          "tmap")

install.packages(pkgs)
```

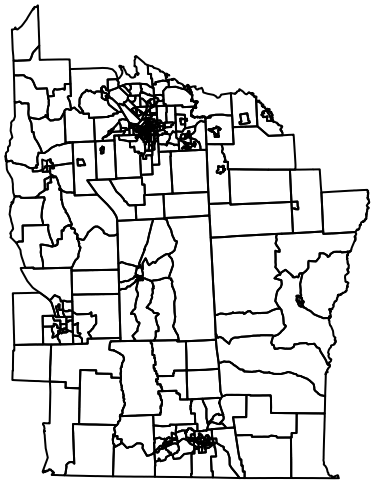
```
library(RColorBrewer)
library(sf)
library(spdep)
library(spatialreg)
library(tmap)
```

2. IMPORTACIÓN DE LOS DATOS

El paquete **sf** servirá para importar y dibujar las geometrías de la base de datos sobre viviendas en Boston y la base de datos sobre Nueva York.

```
boston <- st_read("D:/Docencia_Master_2021/GEOG6000/Sesion03/boston.shp", quiet = TRUE)
plot(st_geometry(boston))
```

```
NY8 <- st_read("D:/Docencia_Master_2021/GEOG6000/Sesion03/NY8_utm18.shp", quiet = TRUE)
plot(st_geometry(NY8))
```



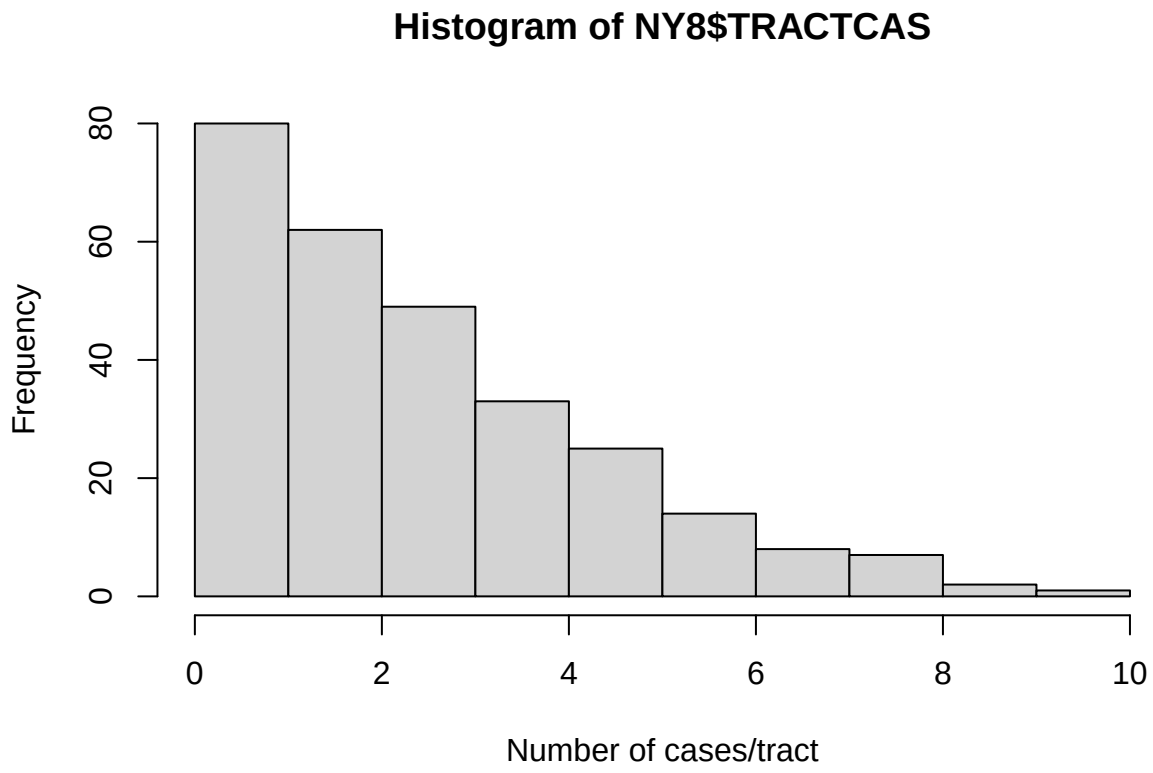
La tabla de atributos asociada a las entidades espaciales puede ser vista tecleando el nombre del objeto `sf` o usando la notación$ para extraer variables individuales:

```
head(NY8)
```

```
## Simple feature collection with 6 features and 17 fields
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: 421423.4 ymin: 4661351 xmax: 427091.6 ymax: 4664822
## Projected CRS: WGS 84 / UTM zone 18N
##          AREANAME     AREAKEY        X        Y POP8 TRACTCAS  PROPCAS
## 1 Binghamton city 36007000100 4.069397 -67.3533 3540     3.08 0.000870
## 2 Binghamton city 36007000200 4.639371 -66.8619 3560     4.08 0.001146
## 3 Binghamton city 36007000300 5.709063 -66.9775 3739     1.09 0.000292
## 4 Binghamton city 36007000400 7.613831 -65.9958 2784     1.07 0.000384
## 5 Binghamton city 36007000500 7.315968 -67.3183 2571     3.06 0.001190
## 6 Binghamton city 36007000600 8.558753 -66.9344 2729     1.06 0.000388
##   PCTOWNHOME PCTAGE65P        Z  AVGIDIST PEXPOSURE   Cases       Xm       Ym
## 1  0.3277311 0.1466102  0.14197 0.2373852  3.167099 3.08284 4069.397 -67353.3
## 2  0.4268293 0.2351124  0.35555 0.2087413  3.038511 4.08331 4639.371 -66861.9
## 3  0.3377396 0.1380048 -0.58165 0.1708548  2.838229 1.08750 5709.063 -66977.5
## 4  0.4616048 0.1188937 -0.29634 0.1406045  2.643366 1.06515 7613.831 -65995.8
## 5  0.1924370 0.1415791  0.45689 0.1577753  2.758587 3.06017 7315.968 -67318.3
## 6  0.3651786 0.1410773 -0.28123 0.1726033  2.848411 1.06386 8558.753 -66934.4
##     Xshift  Yshift                       geometry
```

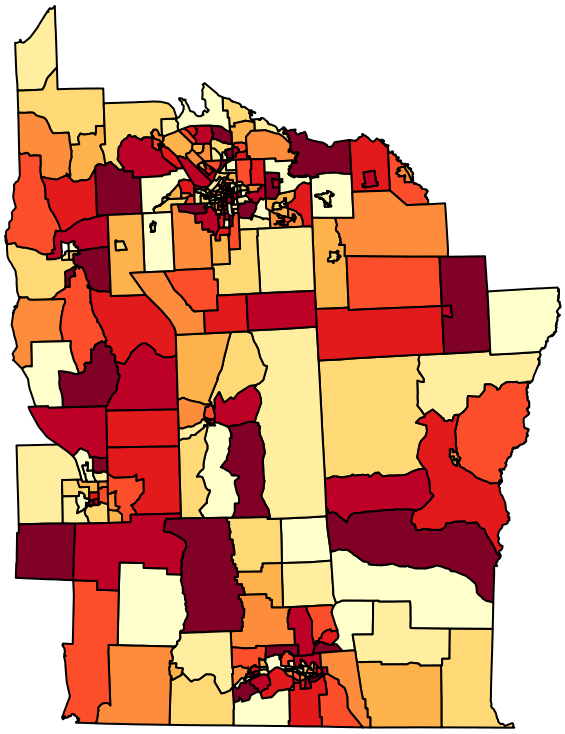
```
## 1 423391.0 4661502 POLYGON ((421840.4 4662874,...
## 2 423961.0 4661993 POLYGON ((421826.4 4663108,...
## 3 425030.6 4661878 POLYGON ((423159 4664759, 4...
## 4 426935.4 4662859 POLYGON ((425261.8 4663493,...
## 5 426637.5 4661537 POLYGON ((425033.3 4662995,...
## 6 427880.3 4661921 POLYGON ((426178.8 4664216,...
```

```
hist(NY8$TRACTCAS,
     xlab = "Number of cases/tract")
```



También es conveniente la elaboración de mapas temáticos especificando la columna que deseamos representar gráficamente:

```
my.pal <- brewer.pal(9, "YlOrRd")

plot(NY8["Cases"],
     main = "Casos de Leucemia en Nueva York",
     col = my.pal)
```



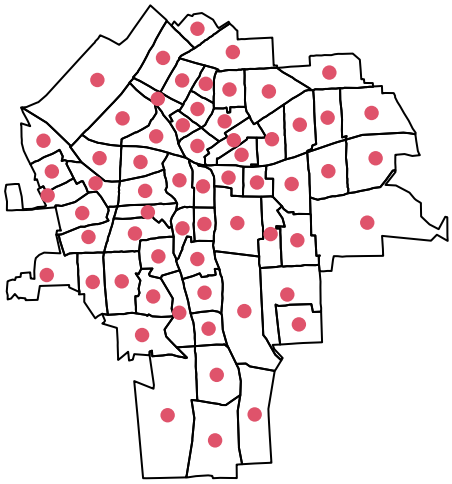
3. CONSTRUCCIÓN DE LA MATRIZ DE PONDERACIONES ESPACIALES.

Con el fin de repasar los diferentes métodos que veremos a continuación, se extraerá el polígono correspondiente a la entidad `Syracuse` que corresponde a un barrio de Nueva York. Posteriormente, obtendremos su geometría usando la función `st_geometry()` y sus centroides con la función `st_centroid()`. Usaremos ambos para visualizar la estructura de vecindad.

```
Syracuse <- NY8[NY8$AREANAME == "Syracuse city", ]
Syracuse.geom <- st_geometry(Syracuse)
Syracuse.coords <- st_centroid(Syracuse.geom)
```

Finalmente, representaremos gráficamente tanto la geometria como los centroides.

```
plot(Syracuse.geom, reset = FALSE)
plot(Syracuse.coords, pch = 16, col = 2, add = TRUE)
```



3.1 Funciones de vecindad

A continuación se enumeran los diferentes métodos para producir estructuras de vecindad en R. Sólo mostraremos gráficamente la primera, aunque el alumno debería representarlos para comprender qué areas son consideradas como vecinas.

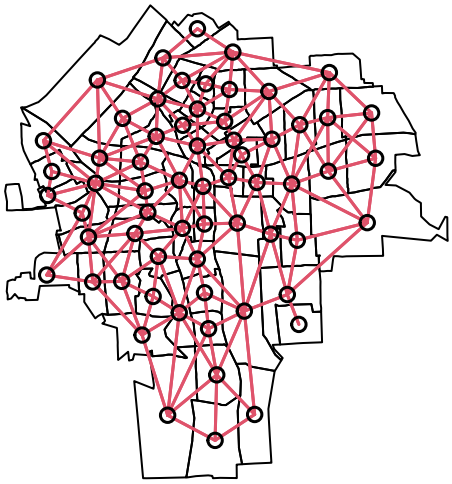
3.1.1 Métodos de límite (Boundary methods)

a. Queen's case Según este método, dos entidades son consideradas vecinas si sus límites o esquinas se tocan; en R se utiliza para ello la función`poly2nb()`:

```
Sy1_nb <- poly2nb(Syracuse)
```

Para visualizar la estructura resultante, primero dibujaremos la geometría de los polígonos originales, y posteriormetne añadiremos la estructura, para lo cual es necesario conocer la vecindad, los centroides y varias opciones de anchura y color de línea:

```
plot(Syracuse.geom, reset = FALSE)
plot(Sy1_nb, Syracuse.coords, add = TRUE, col = 2, lwd = 1.5)
```



b. Rooks's case En este caso, dos regiones son vecinas únicamente si se tocan una parte contígua de sus límites.

```
Sy2_nb <- poly2nb(Syracuse, queen = FALSE)
```

3.1.2 Métodos del Centroide

a. Triangulación de Delaunay Este método construye inicialmente una triangulación entre conjuntos de 3 puntos que se encuetnran dentro de un círculo de radio determinado.

```
Sy3_nb <- tri2nb(Syracuse.coords)
```

b. Esfera de influencia El método de la esfera de influencia restringe las conexiones creadas por la triangulación de Delaunay hasta cierta longitud.

```
Sy4_nb <- graph2nb(soi.graph(Sy3_nb, Syracuse.coords))
```

c. k vecinos más próximos (nearest neighbors) En este método cada región es conectada a sus k vecinos más próximos, independientemente de la distancia existente entre ellos

```
Sy5_nb <- knn2nb(knearneigh(Syracuse.coords, k = 1))
Sy6_nb <- knn2nb(knearneigh(Syracuse.coords, k = 2))
```

d. Funciones de distancia Las funciones de distancia juntan dos regiones si sus centroides están a cierta distancia unos de otros. Esta opción requiere dos parámetros:

- `d1` o distancia mínima
- `d2` o distancia máxima.

De manera arbitraria consideraremos un máximo del 75% de la máxima distancia obtenida cuando utilcemos 2 vecinos más próximos.

Primero, se obtienen las distancias entre los pares de vecinos más próximos (primero se extrae como lista con la función `nbdists()` y posteriormente la convierte en un vector con `unlist()`). En segundo lugar se calcula el máximo valor entre esas distancias. Finalmente, se establece con `d2` el valor máximo de $*0.75$.

```
dists <- nbdists(Sy6_nb, Syracuse.coords)
dists <- unlist(dists) #list to vector

max_1nn <- max(dists)

Sy7_nb = dnearneigh(Syracuse.coords, d1 = 0, d2 = 0.75 * max_1nn)
```

3.1.3 Edición de estructuras de vecindad

Si bien estas funciones brindan una manera rápida de establecer una estructura de vecindad, es probable que incluyan algunas conexiones que no son realistas o que excluyan algunas conexiones reales.

Es posible editar la estructura del vecindad directamente; esta consiste en una lista de n vectores, donde cada vector contiene los vecinos vinculados a un solo polígono. Para ver el contenido:

```
str(Sy1_nb)
```

Para acceder a los vecinos del primer polígono:

```
Sy1_nb[[1]]
```

```
## [1]  2  5 11 21 22
```

Si bien es posible editar las estructuras de vecindad manualmente, no es fácil. Una forma más sencilla es la función `edit.nb ()` que permite la edición interactiva de una estructura de vecindario. Las instrucciones para su uso aparecen en el apéndice de este capítulo.

3.2 Ponderaciones espaciales

El cálculo de ponderaciones espaciales a partir de la función de vecindad se puede realizar de varias formas, que se señalan a continuación.

El primer método no realiza la estandarización (pesos binarios) y el segundo estandariza por filas, de modo que para cualquier área dada, cada vecino es ponderado por el número total de vecinos. La función usada es `nb2listw ()` que devuelve la matriz dispersa que proporciona los enlaces entre vecinos y las ponderaciones de este enlace. Los no vecinos obtienen un peso de cero.

```
Sy1_lw_B <- nb2listw(Sy1_nb, style = 'B')
Sy1_lw_W <- nb2listw(Sy1_nb, style = 'W')
```

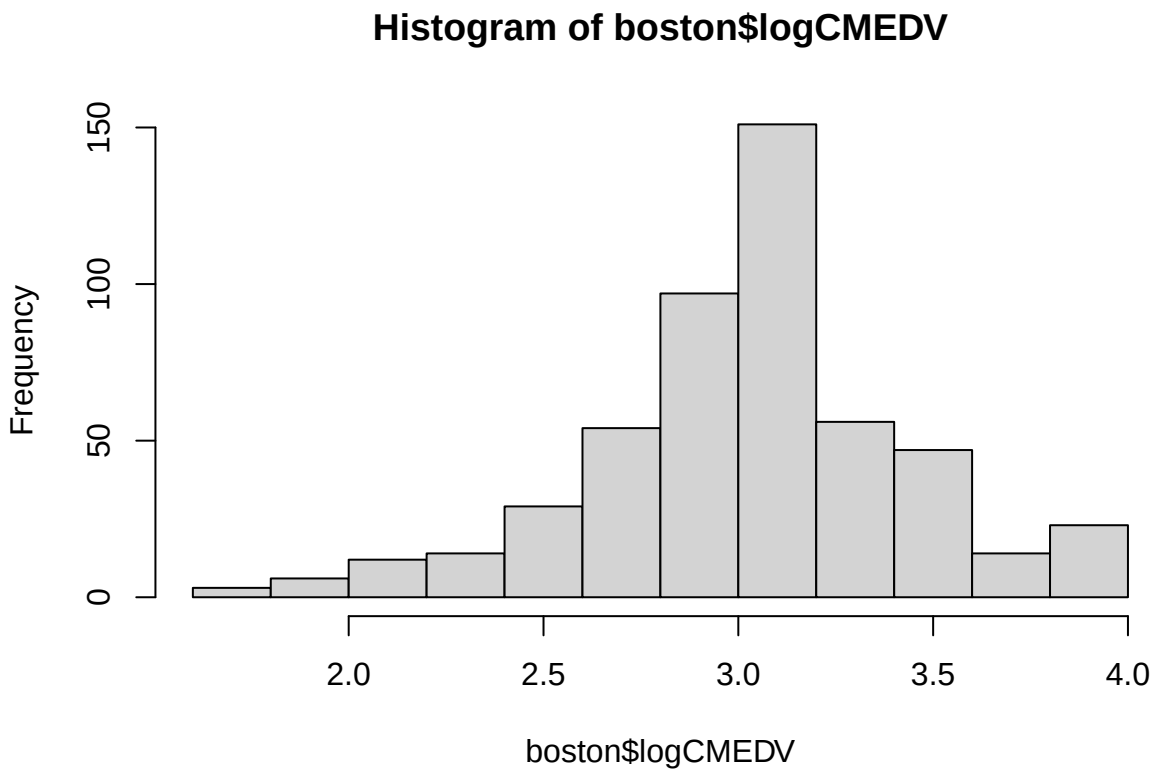
Un método más sofisticado podría incorporar información sobre el conjunto de datos analizados. Por ejemplo, las ponderaciones se pueden asignar mediante un método del inverso de la distancia, donde los vecinos más cercanos tienen pesos más altos. Para realizar esto, es necesario una lista de distancias a lo largo de cada unión para la estructura de vecindad que usaremos. En este caso, usamos la primera de las listas, la generada siguiendo el caso `Queen`. Primero, extraemos las distancias de la lista de vecindad (`nbdists ()`), para obtener una lista de distancias. Luego, generamos distancias inversas dividiendo por 1000 (para hacerlas un poco más manejables) y tomando el recíproco. Esto es un poco complejo ya que obtenemos una salida de lista de `nbdists ()`. Creamos una nueva función, `invd ()`, que calcula la distancia inversa en kilómetros (las coordenadas están en metros). Luego aplicamos esta función a cada elemento de la lista usando `lapply ()`. Esta es la versión * lista * de la función `apply ()` que hemos usado anteriormente.

```
dists <- nbdists(Sy1_nb, Syracuse.coords)

inverse_distance <- function(x) {1/(x/1000)}

idw <- lapply(dists, inverse_distance)

Sy1_lw_idwB <- nb2listw(Sy1_nb, glist = idw, style = "B")
```

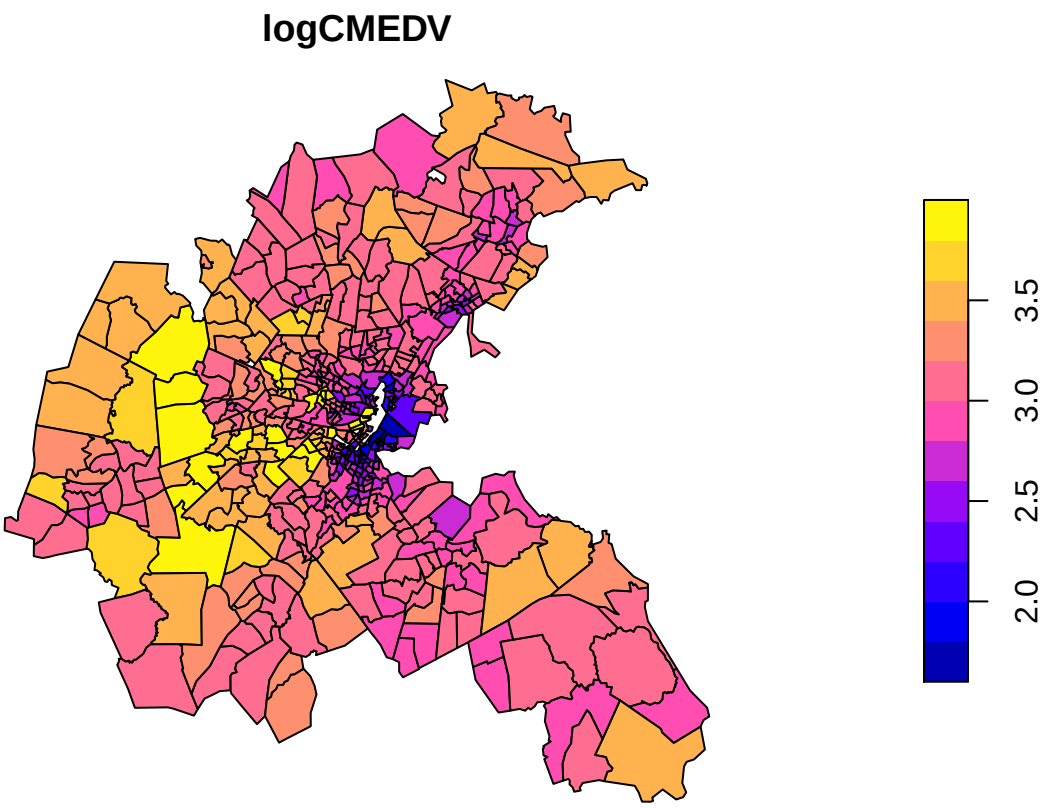
4. AUTOCORRELACIÓN ESPACIAL

A partir de aquí, nos concentraremos en analizar el conjunto de datos de precios de la vivienda en Boston. Los valores medianos del precio de la vivienda que deseamos modelizar muestra una distribución con sesgo (véase histograma), por lo que crearemos una nueva variable que contenga el logarítmo de los valores iniciales (precios de la vivienda):

```
boston$logCMEDV <- log(boston$CMEDV)
hist(boston$logCMEDV)
```

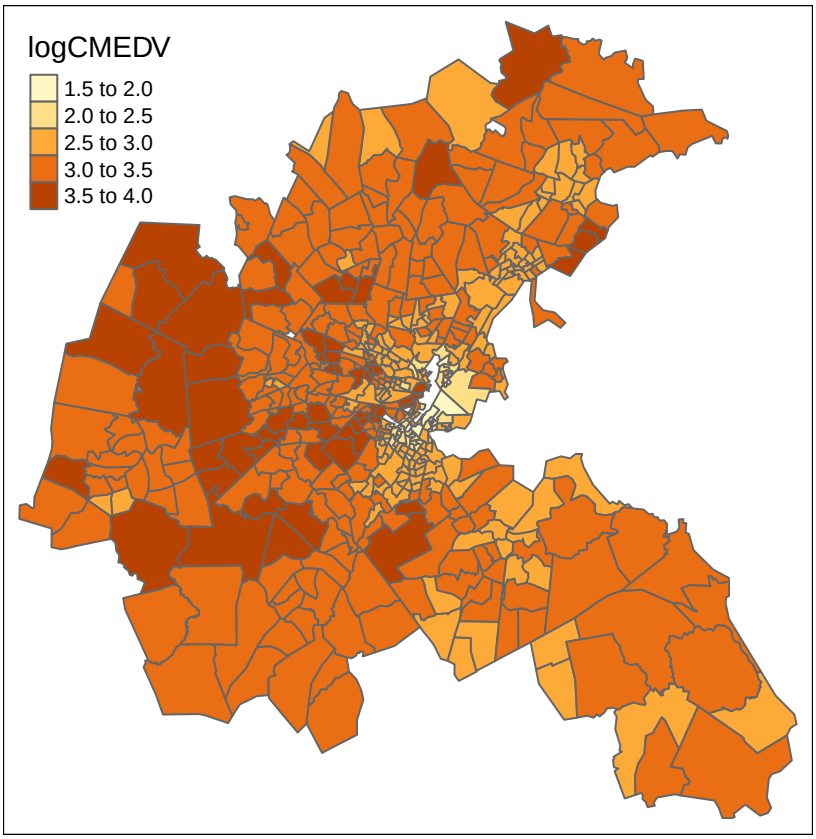



```
plot(boston["logCMEDV"])
```



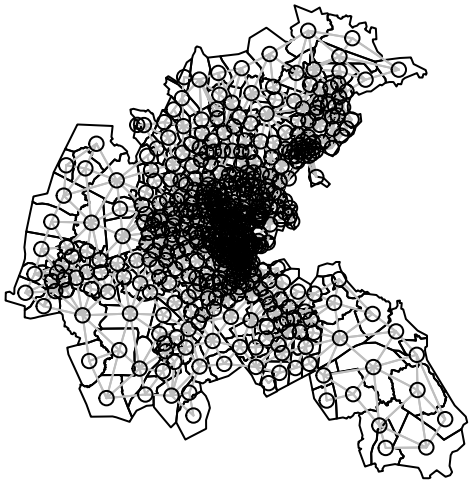
También trazaremos el mapa de esta variable usando **tmap**:

```
tm_shape(boston) +
  tm_fill("logCMEDV") +
  tm_borders()
```



También creamos una estructura de vecindad, utilizando el método de contigüidad de casos de Queen.

```
boston.nb <- poly2nb(boston, queen = TRUE)
coords <- st_centroid(st_geometry(boston))

plot(st_geometry(boston), reset = FALSE)
plot(boston.nb, coords, add = TRUE, col = "gray")
```



Y finalmente convierta esto en una matriz de ponderación espacial.

```
boston.listw = nb2listw(boston.nb)
```

4.1 *I* Global de Moran

Comenzaremos analizando la autocorrelación global en los datos de precios de la vivienda en Boston, aplicando la `moran.test ()`. Estableceremos como verdadero el argumento `randomisation`, ya que no sabemos nada sobre las tendencias espaciales en este conjunto de datos. La función requiere una variable espacial y una matriz de ponderación espacial; como resultados más importantes se encuentran en valor de la I de Moran, la puntuación z (desviación estándar) y el valor de p asociado:

```
moran.test(boston$logCMEDV,
           listw = boston.listw,
           alternative = "two.sided",
           randomisation = TRUE)
```

```
##
##  Moran I test under randomisation
##
## data:  boston$logCMEDV
## weights: boston.listw
##
## Moran I statistic standard deviate = 27.06, p-value < 2.2e-16
```

```
## alternative hypothesis: two.sided
## sample estimates:
## Moran I statistic       Expectation          Variance
##      0.7270399706     -0.0019801980      0.0007258269
```

Simplemente a título comparativo, es posible calcular la I de Moran bajo el supuesto de normalidad, asumiendo que el patrón de precios de la vivienda en Boston es un subconjunto de una tendencia espacial más amplia. El resultado debería ofrecer un pequeño cambio en la varianza calculada, pero no lo suficiente como para afectar nuestra conclusión de que los datos están autocorrelacionados espacialmente.

```
moran.test(boston$logCMEDV,
           listw = boston.listw,
           alternative = "two.sided",
           randomisation = FALSE)
```

```
##
##  Moran I test under normality
##
## data:  boston$logCMEDV
## weights: boston.listw
##
## Moran I statistic standard deviate = 27.038, p-value < 2.2e-16
## alternative hypothesis: two.sided
## sample estimates:
## Moran I statistic       Expectation          Variance
##      0.7270399706     -0.0019801980      0.0007270156
```

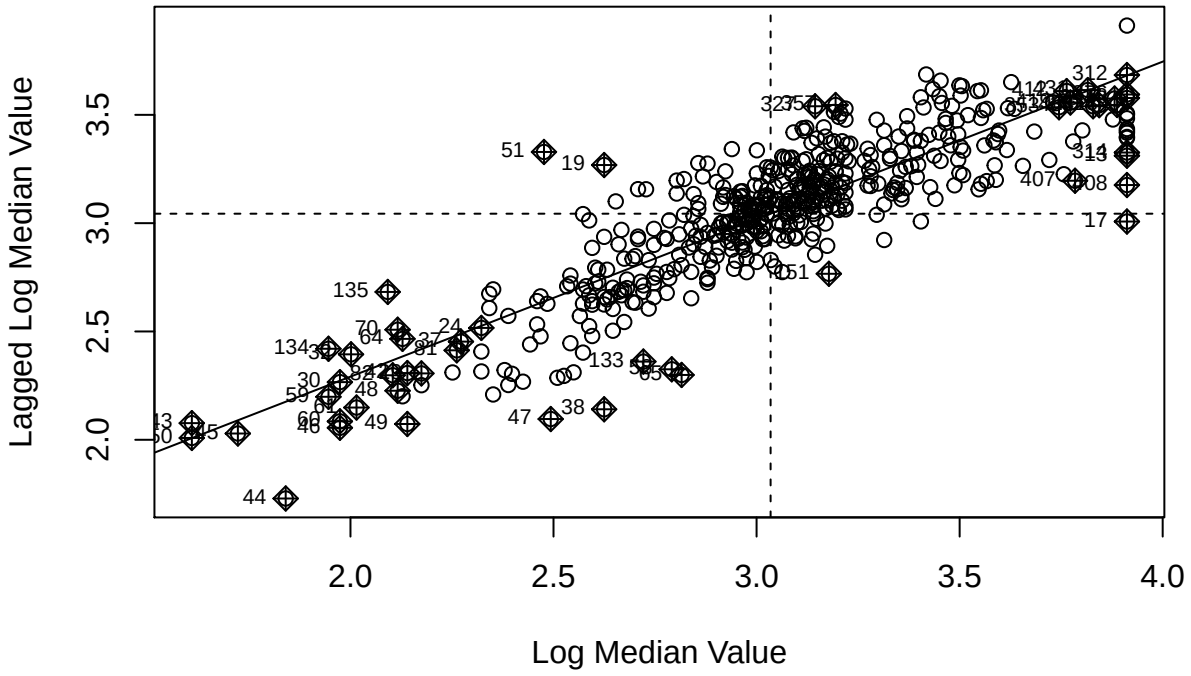
También podemos calcular la importancia de la significación de la autocorrelación observada utilizando un método de Monte Carlo. En este método redistribuimos aleatoriamente los valores en las ubicaciones varios cientos de veces, recalculando el valor de I de Moran cada vez. Una vez hecho esto, comparamos el *rango* de la versión observada del I de Moran con los obtenidos a partir del remuestreo aleatorio. Si estamos en el extremo superior o inferior de todas estas realizaciones aleatorias, es muy probable que la distribución observada esté significativamente autocorrelacionada. Como está basado en un rango, no podemos usar una prueba de dos caras (two-sided test), pero debemos especificar si creemos que la autocorrelación es positiva ('mayor') o negativa ('menor'). El número de simulaciones a ejecutar viene dado por el parámetro `nsim`. Al aumentar, aumentará la precisión del valor de p obtenido (pero tardará más en ejecutarse):

```
moran.mc(boston$logCMEDV,
         listw = boston.listw,
         nsim = 999,
         alternative = 'greater')
```

```
##
##  Monte-Carlo simulation of Moran I
##
## data:  boston$logCMEDV
## weights: boston.listw
## number of simulations + 1: 1000
##
## statistic = 0.72704, observed rank = 1000, p-value = 0.001
## alternative hypothesis: greater
```

A continuación podemos elaborar un diagrama de dispersión de Moran. Éste muestra la relación entre el valor de cualquier área dada y sus vecinos. La pendiente de la línea ajustada es el valor de I de Moran:

```
moran.plot(boston$logCMEDV,
           boston.listw,
           labels = as.character(boston$ID),
           xlab = "Log Median Value",
           ylab = "Lagged Log Median Value")
```



4.2 I Local de Moran

Otro parámetro importante para examinar el patrón espacial de autocorrelación es la I local de Moran. Este devuelve un estadístico (y una puntuación z) para cada área considerada.

```
lm1 = localmoran(boston$logCMEDV,
                 listw = boston.listw,
                 alternative = "two.sided")

head(lm1)
```

```
##             Ii          E.Ii      Var.Ii        Z.Ii Pr(z != E(Ii))
## 1 -0.224510839 -2.873020e-04 0.017914316 -1.67525561    0.093884090
## 2 -0.001798936 -2.175813e-05 0.002735961 -0.03397629    0.972896060
## 3  0.137484323 -9.177414e-05 0.005723568  1.81848429    0.068990146
```

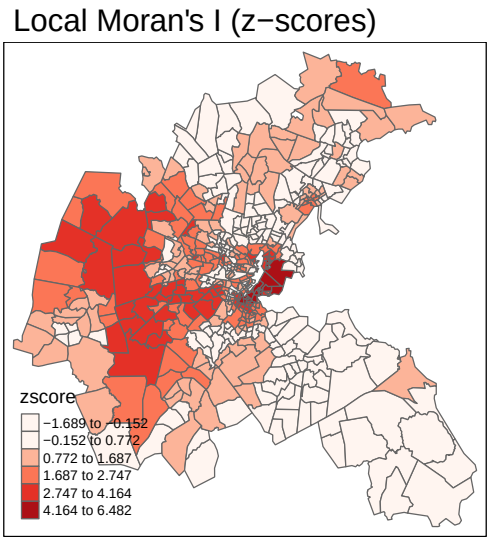
```
## 4  0.033966647 -8.277718e-05 0.010408122  0.33375177    0.738566878
## 5  0.491120158 -4.043873e-04 0.022365621  3.28665963    0.001013833
## 6  0.002533256 -2.287544e-05 0.001909980  0.05848826    0.953359715
```

Los resultados se pueden extraer y cartografiar. Las puntuaciones de z están en la cuarta columna y los valores de p en la quinta columna:

```
boston$zscore <- lm1[,4]
boston$pval <- lm1[,5]
```

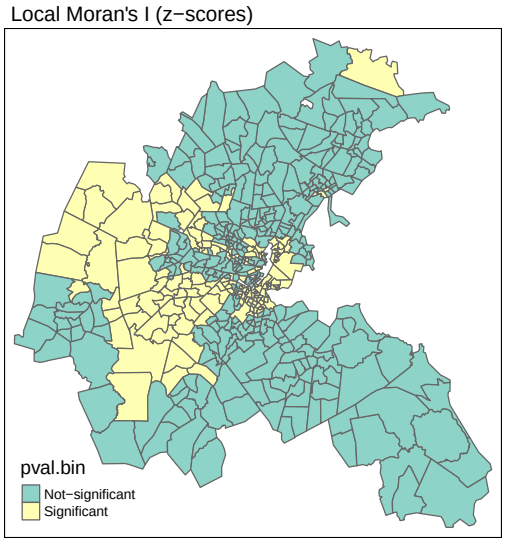
Primero cartografiamos las puntuaciones z:

```
tm_shape(boston) +
  tm_fill("zscore", palette = "Reds", style = "jenks", n = 6) +
  tm_borders() +
  tm_layout(main.title = "Local Moran's I (z-scores)",
            legend.position = c("left", "bottom"))
```



Para los valores de p, los transformaremos en un vector binario: valor 1 en caso de que p sea inferior a 0.05, y cero en caso contrario. Al trazarlos se muestran claramente las áreas con alta autocorrelación, por el puerto y al oeste de la ciudad.

```
boston$pval.bin <- as.factor(ifelse(boston$pval < 0.05, "Significant", "Not-significant"))

tm_shape(boston) +
  tm_fill("pval.bin") +
  tm_borders() +
  tm_layout(main.title = "Local Moran's I (z-scores)",
            main.title.size = 1,
            legend.position = c("left", "bottom"))
```

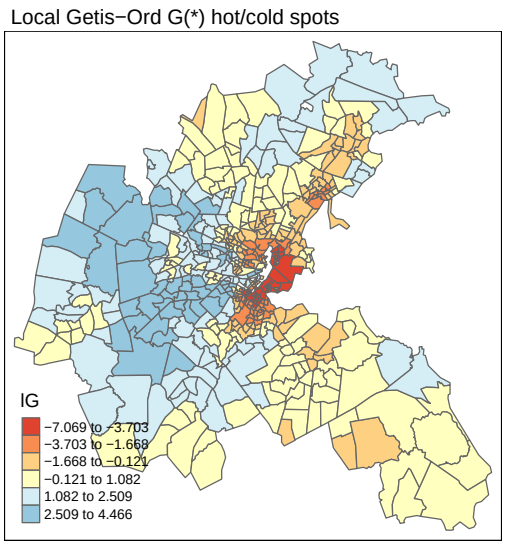


Finalmente, se puede aplicar la estadística Getis-Ord G^* para observar la variación local en los valores, pero identificando regiones con grupos de valores altos o bajos. El estadístico G^* incluye el valor de la ubicación de interés, así como sus vecinos. Para dar cuenta de esto, primero necesitamos hacer una nueva matriz de ponderación espacial que incluya una ponderación para el enlace de una ubicación consigo mismo (con el argumento `include.self ()`). El estadístico se calcula utilizando la función `localG ()`, que toma como entrada la variable de interés y la matriz de ponderación espacial.

```
boston.listwGs <- nb2listw(include.self(boston.nb), style = "B")
boston$lG <- as.numeric(localG(boston$logCMEDV, boston.listwGs))
```

La salida es el estadístico G^*, convertido en puntuaciones z, que también se pueden cartografiar.

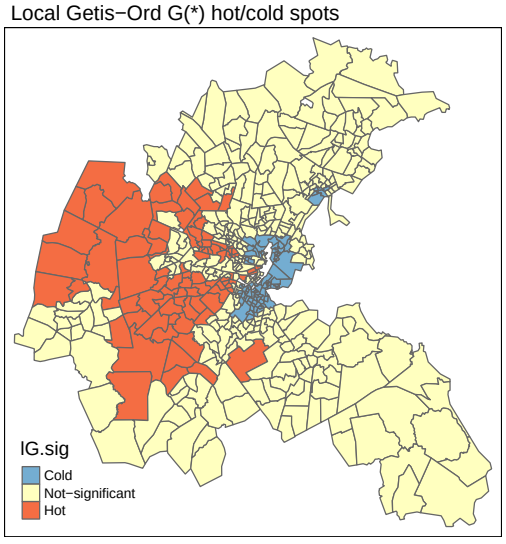
```
tm_shape(boston) +
  tm_fill("lG", palette = "RdYlBu", style = "jenks", n = 6) +
  tm_borders() +
  tm_layout(main.title = "Local Getis-Ord G(*) hot/cold spots",
            main.title.size = 1,
            legend.position = c("left", "bottom"))
```



Los resultados muestran una estructura muy clara con grupos de precios bajos en el centro de Boston, pero interrumpidos por un grupo de precios más altos que corre a lo largo del río hasta el puerto. Otros conglomerados, pero de valores elevados, se encuentran en los suburbios del oeste.

```
boston$lG.sig <- ifelse(boston$lG < -1.96,
                        -1,
                        ifelse(boston$lG > 1.96, 1, 0))

boston$lG.sig <- factor(boston$lG.sig, labels=c("Cold", "Not-significant", "Hot"))
```

```
tm_shape(boston) +
  tm_fill("lG.sig", palette = "-RdYlBu", style = "jenks", n = 6) +
  tm_borders() +
  tm_layout(main.title = "Local Getis-Ord G(*) hot/cold spots",
            main.title.size = 1,
            legend.position = c("left", "bottom"))
```



5. Modelos de regresión espacial

5.1 Importanción y cartografía inicial de los datos.

El paquete **spatialreg** tiene varias funciones para construir modelos de regresión espacial, que en este caso utilizará un conjunto de datos sobre delitos en la ciudad de Columbus.

```
col <- st_read("D:/Docencia_Master_2021/GEOG6000/Sesion03/columbus.shp", quiet = TRUE)
str(col)
```

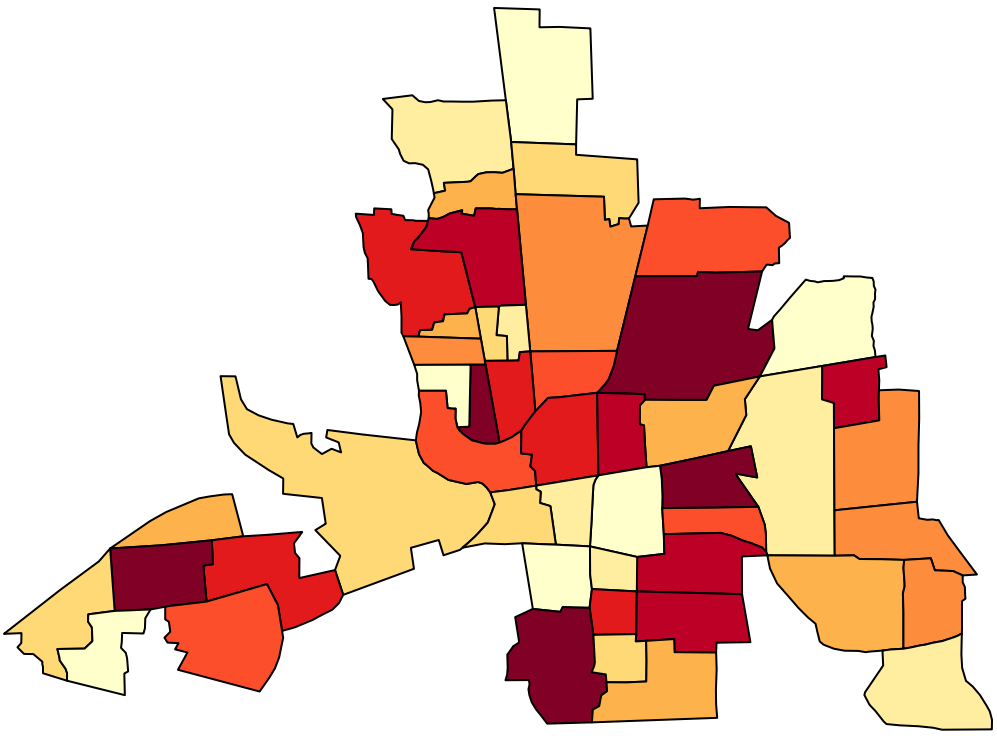
```
## Classes 'sf' and 'data.frame':   49 obs. of  21 variables:
##  $ AREA      : num  0.3094 0.2593 0.1925 0.0838 0.4889 ...
##  $ PERIMETER : num  2.44 2.24 2.19 1.43 3 ...
##  $ COLUMBUS_ : num  2 3 4 5 6 7 8 9 10 11 ...
##  $ COLUMBUS_I: num  5 1 6 2 7 8 4 3 18 10 ...
##  $ POLYID    : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ NEIG      : int  5 1 6 2 7 8 4 3 18 10 ...
##  $ HOVAL     : num  80.5 44.6 26.4 33.2 23.2 ...
##  $ INC       : num  19.53 21.23 15.96 4.48 11.25 ...
##  $ CRIME     : num  15.7 18.8 30.6 32.4 50.7 ...
##  $ OPEN      : num  2.851 5.297 4.535 0.394 0.406 ...
##  $ PLUMB     : num  0.217 0.321 0.374 1.187 0.625 ...
##  $ DISCBD    : num  5.03 4.27 3.89 3.7 2.83 3.78 2.74 2.89 3.17 4.33 ...
##  $ X         : num  38.8 35.6 39.8 36.5 40 ...
##  $ Y         : num  44.1 42.4 41.2 40.5 38 ...
##  $ NSA       : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ NSB       : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ EW        : num  1 0 1 0 1 1 0 0 1 1 ...
##  $ CP        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ THOUS     : num  1000 1000 1000 1000 1000 1000 1000 1000 1000 1000 ...
##  $ NEIGNO    : num  1005 1001 1006 1002 1007 ...
##  $ geometry  :sfc_POLYGON of length 49; first list element: List of 1
##   ..$ : num [1:15, 1:2] 8.62 8.56 8.81 8.81 8.92 ...
##   ..- attr(*, "class")= chr [1:3] "XY" "POLYGON" "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 [1:20] "AREA" "PERIMETER" "COLUMBUS_" "COLUMBUS_I" ...
```

Nuestro objetivo es modelizar la tasa de criminalidad utilizando información sobre el ingreso familiar ("INC") y el precio ("HOVAL"). Como ambas variables presentan sesgos a la derecha, la distribución debe "normalizarse" calculando su logarítmo:

```
col$lINC <- log(col$INC)
col$lHOVAL <- log(col$HOVAL)
```

ATENCIÓN. Cartografie el valor de la criminalidad en la citada ciudad (columna "CRIME") adaptando el código utilizado para cartografiar los datos de Nueva York y Boston.

```
my.pal <- brewer.pal(9, "YlOrRd")

plot(col["CRIME"],
     main = "Tasa de criminalidad",
     col = my.pal)
```



5.2 Construcción de la matriz de ponderación espacial

Primero, crearemos la estructura de vecindad y la matriz de ponderación espacial asociada. El siguiente ejemplo usa la definición de casos `Queen` y pesos binarios, pero estos pueden ser reemplazados por otras opciones.

```
col.geom <- st_geometry(col)
col.coords <- st_centroid(col.geom)
col.nbq <- poly2nb(col)
```

```
plot(col.geom, reset = FALSE)
plot(col.nbq, col.coords, add = TRUE)
```

Ahora convierta esto en una matriz de ponderación espacial:

```
col.listw <- nb2listw(col.nbq)
```

5.3 Análisis de autocorrelation

Use el código dado arriba para buscar la autocorrelación espacial en la variable `CRIME`, usando los estadísticos I de Moran tanto Global como Local.

5.4 Regresión espacial

5.4.1 Regresión lineal (OLS)

En primer lugar, podemos intentar una clásica regresión múltiple, excluyendo toda la información espacial:

```
col.fit1 <- lm(CRIME ~ lINC + lHOVAL, data = col)
summary(col.fit1)
```

```
##
## Call:
## lm(formula = CRIME ~ lINC + lHOVAL, data = col)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -32.361  -5.997  -1.521   7.432  28.013
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  138.074     14.542   9.495 2.06e-12 ***
## lINC         -19.272      5.024  -3.836 0.000379 ***
## lHOVAL       -14.924      4.568  -3.267 0.002059 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.6 on 46 degrees of freedom
## Multiple R-squared:  0.5392, Adjusted R-squared:  0.5191
## F-statistic: 26.91 on 2 and 46 DF,  p-value: 1.827e-08
```

Los resultados del modelo son buenos, con un estadístico F significativo. Sin embargo, dado el patrón en los datos de delitos visualizado en el mapa anterior, es probable que el modelo pueda verse afectado por la autocorrelación, por lo que es necesario probar si existe autocorrelación en los residuos, utilizando para ello el I de Moran:

```
moran.mc(residuals(col.fit1),
         listw = col.listw,
         nsim = 999)
```

```
##
##  Monte-Carlo simulation of Moran I
##
## data:  residuals(col.fit1)
## weights: col.listw
## number of simulations + 1: 1000
##
## statistic = 0.21308, observed rank = 998, p-value = 0.002
## alternative hypothesis: greater
```

En este caso se ha usado la versión Monte Carlo del estadístico I de Moran, pero podría ser reemplazado por otros enfoques, dependiendo de las suposiciones sobre el patrón espacial.

5.4.2 Test multiplicador de Lagrange

La prueba del multiplicador de Lagrange se usa para evaluar si la autocorrelación está en los valores de la variable dependiente o en sus errores, y ayuda a elegir qué modelo de regresión espacial podemos usar.

Primero ejecutamos esta prueba, usando la versión no robusta. Las pruebas vienen dadas por el parámetro `test`; el rango completo se puede encontrar en la página de ayuda de la función.

```
lmt <- lm.LMtests(col.fit1, col.listw, test = c("LMerr","LMlag"))

summary(lmt)
```

```
##  Lagrange multiplier diagnostics for spatial dependence
## data:
## model: lm(formula = CRIME ~ lINC + lHOVAL, data = col)
```

```
## weights: col.listw
##
##       statistic parameter  p.value
## LMerr    4.7913         1 0.028603 *
## LMlag    9.1694         1 0.002461 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Valores altos del estadístico indican que existe una fuente de correlación más probable. Cuando ambos son significativos, se debe utilizar la prueba robusta (`RLMerr` y `RLMlag`) para decidir. Estas pruebas sólidas tienen en cuenta la autocorrelación en un término y luego prueban la autocorrelación restante en el otro término.

```
lmt_robust <- lm.LMtests(col.fit1, col.listw, test = c("RLMerr","RLMlag"))

summary(lmt_robust)
```

```
##  Lagrange multiplier diagnostics for spatial dependence
## data:
## model: lm(formula = CRIME ~ lINC + lHOVAL, data = col)
## weights: col.listw
##
##        statistic parameter p.value
## RLMerr  0.024805         1 0.87485
## RLMlag  4.402809         1 0.03588 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

La evidencia de una autocorrelación significativa se muestra en la prueba "RLMlag", lo que sugiere un modelo de retraso espacial.

5.4 Modelo de retardo espacial

Un modelo de retardo espacial se puede ajustar a los datos usando la función `lagsarlm ()`. La sintaxis es similar a la mayoría de funciones de modelización en R, excepto que debe proporcionarse una matriz de ponderación espacial:

```
col.fit2 <- lagsarlm(CRIME ~ lINC + lHOVAL,
                     data = col,
                     col.listw)

summary(col.fit2)
```

```
##
## Call:lagsarlm(formula = CRIME ~ lINC + lHOVAL, data = col, listw = col.listw)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -36.1856  -5.4709  -0.5156   6.4413  22.5983
##
## Type: lag
## Coefficients: (asymptotic standard errors)
```

```
##             Estimate Std. Error z value  Pr(>|z|)
## (Intercept)  99.5700    16.1538  6.1639 7.099e-10
## lINC        -11.9221     4.5544 -2.6177 0.0088524
## lHOVAL      -13.5984     3.9965 -3.4026 0.0006676
##
## Rho: 0.42092, LR test value: 8.9447, p-value: 0.0027828
## Asymptotic standard error: 0.12671
##     z-value: 3.3219, p-value: 0.00089398
## Wald statistic: 11.035, p-value: 0.00089398
##
## Log likelihood: -183.6196 for lag model
## ML residual variance (sigma squared): 100.73, (sigma: 10.036)
## Number of observations: 49
## Number of parameters estimated: 5
## AIC: 377.24, (AIC for lm: 384.18)
## LM test for residual autocorrelation
## test value: 0.028921, p-value: 0.86496
```

En la salida de los datos hay que señalar varios aspectos:

- Estimaciones de los coeficientes asociados a las variables independientes.
- El valor del coeficiente `rho`, que muestra la fuerza y la importancia del componente espacial autorregresivo.
- La prueba LM de residuos para buscar la autocorrelación restante.
- El AIC y la probabilidad logarítmica que dan una estimación de la bondad de ajuste del modelo.

Al igual que con el modelo anterior, podemos probar los residuales para cualquier autocorrelación restante:

```
moran.mc(residuals(col.fit2),
         listw = col.listw,
         nsim = 999)
```

```
##
##  Monte-Carlo simulation of Moran I
##
## data:  residuals(col.fit2)
## weights: col.listw
## number of simulations + 1: 1000
##
## statistic = 0.010517, observed rank = 641, p-value = 0.359
## alternative hypothesis: greater
```

5.5 Modelo de error espacial

Si bien los resultados de la prueba del multiplicador de Lagrange indicaron un modelo de retardo espacial como el método más adecuado, también podemos ajustar un modelo de error espacial, utilizando la función `errorsarlm ()`:

```
col.fit3 = errorsarlm(CRIME ~ lINC + lHOVAL,
                      data = col,
                      col.listw)

summary(col.fit3)
```

```
##
## Call:errorsarlm(formula = CRIME ~ lINC + lHOVAL, data = col, listw = col.listw)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -33.3722  -6.3766  -0.1172   6.9844  21.9846
##
## Type: error
## Coefficients: (asymptotic standard errors)
##             Estimate Std. Error z value  Pr(>|z|)
## (Intercept) 117.7761    14.7937  7.9612 1.776e-15
## lINC         -9.9844     4.9172 -2.0305  0.042305
## lHOVAL      -16.0041     4.1761 -3.8323  0.000127
##
## Lambda: 0.53456, LR test value: 6.9048, p-value: 0.0085966
## Asymptotic standard error: 0.14043
##     z-value: 3.8066, p-value: 0.00014091
## Wald statistic: 14.49, p-value: 0.00014091
##
## Log likelihood: -184.6396 for error model
## ML residual variance (sigma squared): 101.71, (sigma: 10.085)
## Number of observations: 49
## Number of parameters estimated: 5
## AIC: 379.28, (AIC for lm: 384.18)
```

En la salida de la función, observe el valor de `lambda`, el coeficiente autorregresivo que representa la fuerza de la autocorrelación en los residuos de un modelo lineal. Tenga en cuenta el AIC y compárelo con el modelo anterior.

5.6 Modelo de retardo spacial de Durbin

Hasta ahora, solo hemos considerado la correlación entre los valores de la variable dependiente en cualquier zona y sus vecinos. Podría surgir una fuente alternativa de dependencia espacial entre los valores de una variable dependiente y los valores vecinos de una variable independiente.

Para modelizar esto, podemos usar un modelo espacial de Durbin, que incluye variables independientes con retardo en las zonas vecinas. Este procedimiento usa la función `lagsarlm ()`, pero con el parámetro `type` establecido en `mixed`, para especificar un modelo de retardo espacial de Durbin:

```
col.fit4 = lagsarlm(CRIME ~ lINC + lHOVAL,
                    data = col,
                    col.listw,
                    type = 'mixed')

summary(col.fit4)
```

```
##
## Call:lagsarlm(formula = CRIME ~ lINC + lHOVAL, data = col, listw = col.listw,
##     type = "mixed")
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -36.42025  -5.66457   0.10208   5.61973  21.82351
```

```
##
## Type: mixed
## Coefficients: (asymptotic standard errors)
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  95.4310    31.8319  2.9980 0.002718
## lINC         -9.6490     4.9167 -1.9625 0.049702
## lHOVAL      -15.2748     4.1030 -3.7229 0.000197
## lag.lINC    -13.6313     8.4749 -1.6084 0.107743
## lag.lHOVAL   11.8547     8.2710  1.4333 0.151776
##
## Rho: 0.35363, LR test value: 3.4569, p-value: 0.062987
## Asymptotic standard error: 0.16701
##     z-value: 2.1174, p-value: 0.034225
## Wald statistic: 4.4834, p-value: 0.034225
##
## Log likelihood: -182.0123 for mixed model
## ML residual variance (sigma squared): 95.663, (sigma: 9.7807)
## Number of observations: 49
## Number of parameters estimated: 7
## AIC: 378.02, (AIC for lm: 379.48)
## LM test for residual autocorrelation
## test value: 0.14333, p-value: 0.705
```

La salida proporciona los coeficientes para todas las variables, incluidas las versiones retrasadas. ¿Alguno de estos es significativo?

6. Ejercicio

1. El archivo *usa48_usedcars.shp* contiene información sobre las tasas impositivas y los gastos de envío de los automóviles nuevos ("tax_charge") en los 48 estados contiguos de EE. UU. Durante el período 1955-1959, así como el promedio de automóviles usados precio de 1960 (`price_1960`). Utilice este conjunto de datos para crear un modelo espacial que vincule los precios de los automóviles usados con los impuestos y los costos de envío (paquete **spdep**).

    - Construya una función de vecindad que vincule los 48 estados. Deberá elegir una de las funciones de vecindario disponibles en R. Explique su elección de función de vecindario
    - Construya una matriz de ponderación espacial. Utilice la función `summary ()` para obtener información sobre la distribución de enlaces por estado e informar de esto
    - Utilice la función `moran.test ()` o `moran.mc ()` para probar la autocorrelación espacial en los precios de los coches usados. Extrae el estadístico I de Moran, la puntuación z y si puede rechazar o no la hipótesis nula de que no hay autocorrelación espacial.
    - Construya un modelo lineal simple usando la función `lm ()` entre los precios de los autos usados (variable dependiente o variable Y) y el costo de impuestos y envío (variables independiente o variable X). Extraiga los los coeficientes del modelo y el R^2. Verifique la autocorrelación en los residuos del modelo usando la función `moran.mc ()`, e indique si está presente o no
    - Utilice la prueba del multiplicador de Lagrange para identificar si se debe utilizar un modelo de error espacial o de retraso espacial. Recuerde que es posible que deba utilizar la versión robusta de esta prueba si los resultados no robustos son significativos. Informe el valor p de cada prueba y dé la opción del modelo
    - Ahora cree un modelo espacial que vincule los precios del automóvil y el costo de impuestos/envío, utilizando el modelo que eligió en la sección anterior (use la función `lagsarlm ()` o `errorsarlm ()`). Señale la siguiente información:

  - Si se utiliza un modelo de retardo espacial: a) coeficientes (y su importancia); b) el valor de Rho (el coeficiente autorregresivo espacial); c) el valor AIC y el valor AIC del modelo lineal sin la matriz de ponderación espacial
  - Si se usa un modelo de error espacial: a) coeficientes (y su importancia); b) el valor de lambda (el coeficiente autorregresivo espacial); c) el valor AIC y el valor AIC del modelo lineal sin la matriz de ponderación espacial
- Pruebe la autocorrelación remanente en los residuos del modelo usando la función `moran.test ()` o `moran.mc ()`. Dado el valor que obtiene, indique si cree que el modelo tiene en cuenta adecuadamente la autocorrelación.
- ¿Es significativo el coeficiente que relaciona impuestos y entrega con el precio del automóvil? Si no es así, dé una razón por la que este puede no ser el caso.

7. Apéndice: Edición de una vecindad

El paquete **spdep** incluye una función para la edición interactiva de estructuras de vecindad. Sin embargo, no funciona actualmente en RStudio debido a problemas ligados al cambio de tamaño de la ventana gráfica.

En su lugar, deberá ejecutar esto desde la aplicación R base. Como ejemplo, abra R (¡no RStudio!) desde el menú de su aplicación o el menú de inicio. Una vez abierto, use el menú 'Misc' y elija 'Cambiar directorio de trabajo' para buscar los archivos de laboratorio. Una vez allí, cargue el conjunto de datos de NY, cree el subconjunto de Syracuse y cree una estructura de vecindario:

```
library(sf)
library(spdep)
library(sp)

NY8 <- st_read("D:/Docencia_Master_2021/GEOG6000/Sesion03/NY8_utm18.shp")
Syracuse <- NY8[NY8$AREANAME == "Syracuse city", ]
Sy1_nb <- poly2nb(Syracuse)
```

Tenga en cuenta que cuando crea un gráfico en la base R, se abre una ventana separada para mostrar esto. Ahora, editemos esta estructura. Esta función aún no funciona con objetos `sf`, por lo que primero necesitaremos convertir Syracuse usando la biblioteca ** sp **. Ejecute el siguiente código:

```
Syr2 <- as_Spatial(Syracuse)
coords <- coordinates(Syr2)
Sy2_nb <- edit.nb(Sy1_nb, coords, polys = Syr2)
```

Esto mostrará los polígonos, centroides y estructura de vecindad, y la consola mostrará `Identificando contigüidad para eliminación`. Ahora puede hacer dos cosas:

- Si selecciona dos centroides que ya están vinculados, la función le preguntará si desea `Eliminar esta línea (y / n)`. Ingrese `y` para hacerlo, o `n` para ignorar esto y continuar
- Si selecciona dos centroides que no están vinculados, la función le preguntará si desea `Agregar contigüidad (y/n)`. Ingrese `y` para hacerlo, o`n` para ignorar esto y continuar

Cada vez que agregue o elimine un enlace, la función le preguntará si desea actualizar, continuar o salir. Continuar simplemente espera a que seleccione dos centroides más. Si actualiza, volverá a dibujar la estructura del vecindario resaltando qué enlaces se han eliminado (línea discontinua) o agregado (línea amarilla). Si sale, la sesión interactiva terminará y se escribirá la nueva estructura de vecindario. En el código anterior, asignamos la salida de la función a `Sy2_nb`. Si ahora traza este y el anterior, debería ver las ediciones

que ha realizado. Esta nueva estructura de vecindad se puede utilizar en todas las funciones posteriores (I de Moran, regresión espacial, etc.). Para mantener y reutilizar esta estructura en RStudio, asegúrese de guardarla. Como el objeto de vecindad es algo complicado, la forma más sencilla de hacerlo es escribirlo en un archivo binario R (esto guarda ambas estructuras de vecindad en un solo archivo):

```
save(Sy1_nb, Sy2_nb, file="Sy_nb.RData")
```

You can then load this in a new R or RStudio session, which will recreate these objects

```
load(file="Sy_nb.RData")
```