M1683. Intensificación en Saberes Técnicos: Diseño y Gestión de Bases de Datos Territoriales. Regresion Espacial II

Domingo Rasilla

03 enero, 2022

Contents

0 INTRODUCCIÓN

En este segundo apartado, cubriremos algunos de los métodos más avanzados para la modelización de datos espaciales. Aunque los ejemplos harán uso de datos de naturaleza contínua, estos métodos se pueden extender a otros tipos de datos.

Los archivos que se utilizarán son los siguientes:

- Ratios de crímenes en Columbus, Ohio (*columbus.zip*).
- Precios de la vivienda en Boston (*boston.tr.zip*).
- Índices de probreza según condados en el SE de EEUU (**)

Para llevar a cabo los ejercicios, se necesita activar paquetes de diferentes naturaleza: paquetes para el análisis de datos espaciales (**sf**, **spdep**, **espacialreg** y **spgwr**) y paquetes para cartografiar los resultados **tmap**) para mapear algunos de los resultados, pero puedes usar ** ggplot2 ** o el comando básico `plot ()`.

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

install.packages(pkgs)
```

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

1. Base de datos tasa de crimen en la ciudad de Columbus.

1.1 Importación de los datos

El fichero de datos está disponible como shapefile en *columbus.zip*.

```
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" ...
```

Como señalamos en una sesión anterior, las variables HOVAL (valor de la vivienda, en $1,000) y INC (ingresos por unidad familiar) en $1,000) presentan una distribución no normal. Por ello, ambas deben ser transformadas en su logarítmo.

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

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

1.2 Construcción de la matriz de ponderaciones espaciales

En segundo lugar, se creará la estructura de vecindad y la matriz de ponderación espacial asociada. En ese ejemplo se utiliza el modo Reina y pesos binarios, pero esta elección puede ser reemplazada por otras opciones.

```
col.nbq <- poly2nb(col)

col.nbq
```

```
## Neighbour list object:
## Number of regions: 49
## Number of nonzero links: 236
## Percentage nonzero weights: 9.829238
## Average number of links: 4.816327
```

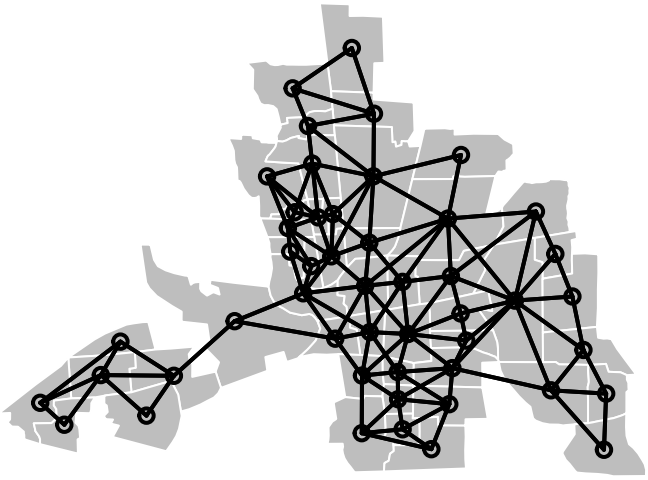
```
col.listw <- nb2listw(col.nbq)
```

```
col.geom <- st_geometry(col)

plot(col.geom,
     col = "gray",
     border = "white")

coords <- st_coordinates(st_centroid(col))

plot(col.nbq, coords,
     add = TRUE,
     col = 1,
     lwd = 2)
```



2. FILTRADO ESPACIAL

El filtrado espacial es un enfoque de modelización espacial introducido por Griffith (1978). Utiliza un eigenanálisis de la matriz de ponderación espacial (la matriz que describe qué ubicaciones se consideran vecinas) para generar patrones espaciales (denominados vectores propios de Moran) que potencialmente representan la dependencia espacial en los datos.

Usando un enfoque paso a paso, es posible ver cuál de estos patrones (o conjunto de patrones) representa el error autocorrelacionado. Luego, estos pueden usarse como variables adicionales en el modelo para *filtrar* la autocorrelación, sin afectar los coeficientes y sus errores estándar estimados de manera robusta.

Para un modelo de regresión espacial, estos patrones se generan a partir del producto de la matriz de ponderación espacial (W) y la matriz de variables independientes (X); el resultado son patrones espaciales vinculados a la distribución de las variables. Por lo tanto, el filtro puede informar de una potencial autocorrelación de las variables utilizadas en el modelo.

En R, el filtrado espacial se realiza como una serie de pasos:

- Construcción de un modelo de regresión por mínimos cuadrados (OLS) entre las variables dependientes e independientes
- Ejecutar el procedimiento paso a paso para la selección de los autovectores de Moran (función `SpatialFiltering ()`) qué se usarán para filtrar los datos. El vector propio seleccionado en cada paso es el que reduce más los valores del estadístico I de Moran para el modelo (es decir, filtra la mayor autocorrelación espacial).
- Reconstrucción del model incluyendo el conjunto seleccionado de autovectores de Moran.

En nuestro caso, aplicaremos el filtro espacial al conjunto de datos sobre delincuencia en Columbus. Comenzamos construyendo el modelo OLS y ejecutamos una prueba I de Moran en sus residuos:

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

summary(lm.ols)
```

```
##
## 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
```

```
moran.test(residuals(lm.ols), col.listw)
```

```
##
##  Moran I test under randomisation
##
## data:  residuals(lm.ols)
## weights: col.listw
##
## Moran I statistic standard deviate = 2.5465, p-value = 0.00544
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##       0.213076117      -0.020833333       0.008437311
```

En segundo lugar, se construirá el filtro espacial. Para ello es necesario suministrar al programa los siguientes parámetros:

- La estructura del vecindario (**no** la matriz de ponderación espacial).
- Los pesos a utilizar (`estilo`). Lo configuramos aquí como `style = "C"`, que es una estandarización global. Para otras opciones, consulte la ayuda de `nb2listw ()`.
- La "regla de detención" (`alpha`). Los vectores propios de Moran se seleccionan hasta que el valor p de la prueba I de Moran por pasos supere este valor.
- Argumento `ExactEV = TRUE` calcula valores exactos para el valor esperado y la varianza de la I de Moran. Si se establece como FALSE, los valores se calcularán utilizando una aproximación rápida. Esto puede reducir en gran medida la cantidad de tiempo necesario, especialmente con grandes conjuntos de datos.

```
sf.err <- SpatialFiltering(lm.ols,
                           nb = col.nbq,
                           style = "C",
                           alpha = 0.25,
                           ExactEV = TRUE,
                           data = col)
```

La salida de la función `SpatialFiltering ()` es un objeto que contiene información sobre el proceso paso a paso y los vectores propios de Moran seleccionados. La información resumida se puede obtener escribiendo:

```
sf.err
```

```
##   Step SelEvec      Eval      MinMi   ZMinMi      Pr(ZI)        R2     gamma
## 0    0       0 0.0000000 0.21262754 2.965794 0.003019023 0.5391541   0.00000
## 1    1       4 0.6711982 0.12538574 2.132225 0.032988341 0.6128150 -31.46222
## 2    2       3 0.7999718 0.01474576 1.053091 0.292299441 0.6673701  27.07627
```

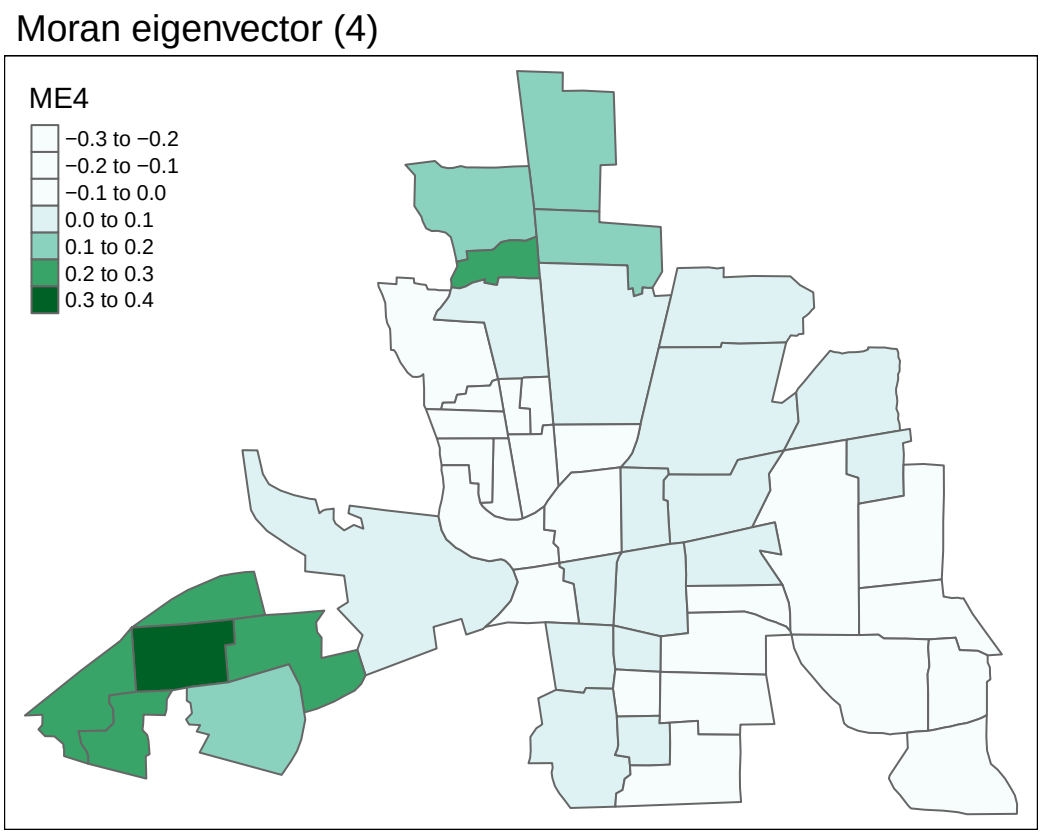
Cada fila de la tabla es un paso y las columnas incluyen la siguiente información:

- `Step`: el número de paso
- `SelEvec`: el valor propio del vector propio seleccionado
- `MinMi`: I de Moran para los residuos con ese vector propio (y los vectores propios anteriores) incluidos como filtro
- `ZMinMi`: I de Moran para los residuos transformados en puntuaciones z.
- `Pr (ZI)`: el valor p para la puntuación z.
- `R2`: el valor R^2 del modelo con autovectores incluidos.
- `gamma`: coeficiente de regresión del vector propio seleccionado en ajuste.

Podemos trazar vectores propios de Moran individuales para examinar el patrón espacial que representan. El primer vector propio seleccionado es el 4:

```
col$ME4 = sf.err$dataset[,"vec4"]

tm_shape(col) +
  tm_fill("ME4", palette = "BuGn") +
  tm_borders() +
  tm_layout(main.title = "Moran eigenvector (4)", main.title.size = 1.2)
```



Ahora podemos aplicar el filtro espacial al modelo `lm ()` original. Primero extraemos los valores para los autovectores seleccionados usando "ajustado", luego usamos esto como un término en el modelo.

Como hay más de un vector seleccionado, esto forma una matriz. Esto se puede incluir en la llamada a `lm ()` directamente, pero cada vector se usará como una covariable individual en la nueva regresión:

```
E.sel <- fitted(sf.err)

lm.sf <- lm(CRIME ~ INC + HOVAL + E.sel,
            data = col)

summary(lm.ols)
```

```
##
## 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
```

```
summary(lm.sf)
```

```
##
## Call:
## lm(formula = CRIME ~ INC + HOVAL + E.sel, data = col)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -34.105  -6.579   0.048   5.541  19.203
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  68.54643    4.11323  16.665  < 2e-16 ***
## INC          -1.57155    0.29052  -5.409 2.47e-06 ***
## HOVAL        -0.28168    0.08962  -3.143  0.00299 **
## E.selvec4   -33.15266    9.92734  -3.340  0.00172 **
## E.selvec3    24.12951    9.94142   2.427  0.01938 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.925 on 44 degrees of freedom
## Multiple R-squared:  0.6774, Adjusted R-squared:  0.6481
## F-statistic:  23.1 on 4 and 44 DF,  p-value: 2.459e-10
```

Los coeficientes no han cambiado con respecto al modelo de regresión original, pero su significación ha mejorado a medida que se ha filtrado la autocorrelación en el término de error. Además, la contabilización de la variación adicional ha aumentado los R^2.

Finalmente, volvemos a ejecutar la prueba I de Moran en los residuos del modelo para verificar que se haya tenido en cuenta la autocorrelación:

```
moran.test(residuals(lm.sf), col.listw)
```

```
##
##  Moran I test under randomisation
##
## data:  residuals(lm.sf)
## weights: col.listw
##
## Moran I statistic standard deviate = 0.69056, p-value = 0.2449
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##        0.04163638       -0.02083333        0.00818341
```

Como estos son dos modelos lineales anidados, podemos usar `anova ()` para probar si la complejidad adicional en el modelo SF ha resultado en un mejor modelo general:

```r
anova(lm.ols, lm.sf)
```

```
## Analysis of Variance Table
##
## Model 1: CRIME ~ lINC + lHOVAL
## Model 2: CRIME ~ INC + HOVAL + E.sel
##   Res.Df    RSS Df Sum of Sq      F  Pr(>F)
## 1     46 6192.9
## 2     44 4334.6  2    1858.4 9.4321 0.00039 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

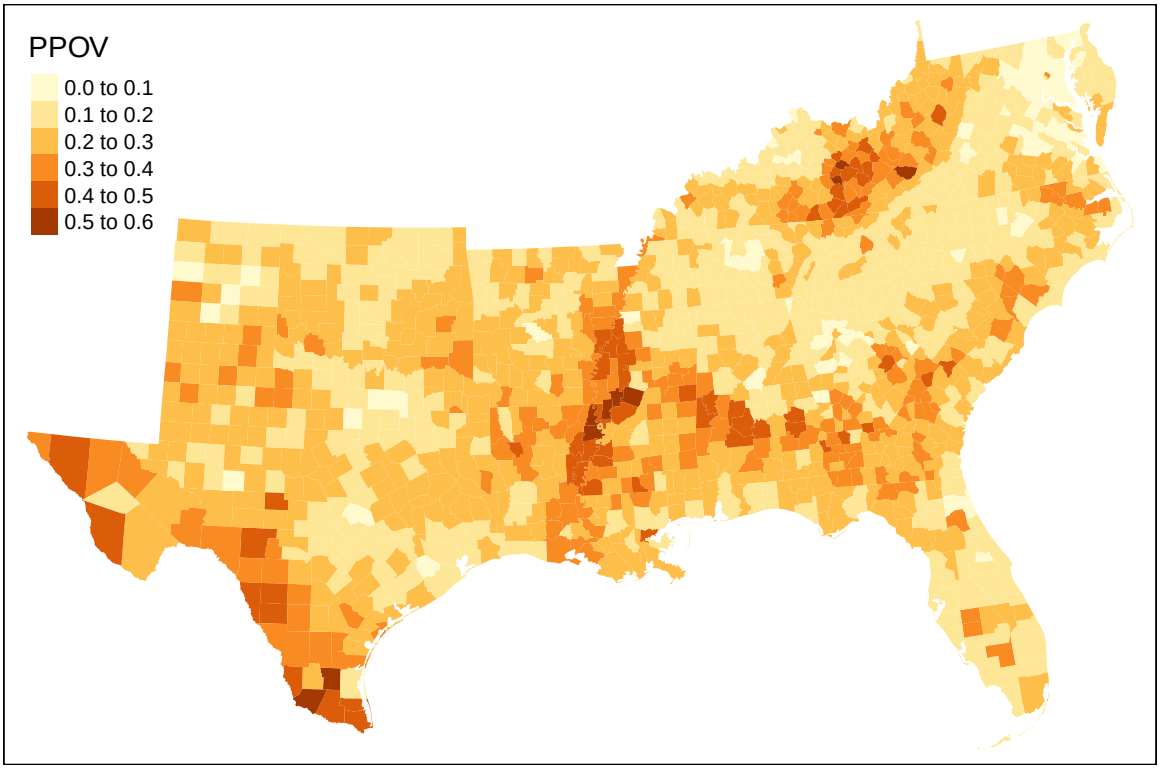
Y el bajo valor de p sugiere que vemos una mejora significativa en el ajuste.

3. REGRESIÓN PONDERADA GEOGRÁFICAMENTE

Fotheringham y otros introdujeron la regresión ponderada geográficamente (GWR) para tratar los problemas de autocorrelación y heterogeneidad en datos espaciales. En lugar de intentar construir un modelo global único para una región de estudio, este procedimiento construye muchos modelos locales dentro de una pequeña ventana del área total. Por ello, el tamaño de la ventana es importante, ya que dicta el número de observaciones utilizadas en cada modelo local y, por lo tanto, la calidad de ese modelo. Si bien ha habido varios artículos que critican este enfoque como método de modelado completo, es muy útil para explorar la variación potencial en la relación entre las variables dependientes e independientes, así como posibles especificaciones erróneas (variables independientes faltantes). Lo usaremos aquí para explorar un conjunto de datos de las tasas de pobreza infantil en el sureste de EE. UU, que se encuentran en el archivo *south00.shp*.

```r
south.sf <- st_read("D:/Docencia_Master_2021/GEOG6000/Sesion03/south00.shp", quiet = TRUE)

## Mapa con el porcentaje de pobres en cada condado
tm_shape(south.sf) +
  tm_fill("PPOV") +
  tm_layout(main.title = "Southern US poverty rates")
```



Lo primero que debemos realizar es crear una estructura de vecindad para el análisis de autocorrelación:

```
south_nb <- poly2nb(south.sf, queen = TRUE)

south_listw <- nb2listw(south_nb, style = "W")
```

Posteriormente, se obtiene el I de Moran para verificar el grado y la importancia de la autocorrelación en la variable de pobreza (`PPOV`).

```
moran.test(south.sf$PPOV, listw = south_listw)
```

```
##
##  Moran I test under randomisation
##
## data:  south.sf$PPOV
## weights: south_listw
##
## Moran I statistic standard deviate = 36.544, p-value < 2.2e-16
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##      0.5892974827     -0.0007215007      0.0002606743
```

A continuación, construimos un modelo de regresión según mínimos cuadrados, utilizando la raíz cuadrada de los porcentajes de pobreza como variable dependiente y las siguientes variables como variables independientes:

- `PFHH`: Porcentaje de mujeres cabezas de familia.
- `PUNEM`: Porcentaje de desempleados.
- `PBLK`: Porcentaje de población negra.
- `P65UP`: Porcentaje de personas mayores de 65 años.
- `METRO`: Área metropolitana (binario).
- `PHSPLUS`: Porcentaje con educación más allá de la escuela secundaria.

Una vez que se construye el modelo, verificamos la autocorrelación en los residuos con `moran.test ()`:

```
fit1 <- lm(SQRTPPOV ~ PFHH + PUNEM + PBLK + P65UP + METRO + PHSPLUS,
           data = south.sf)

summary(fit1)
```

```
##
## Call:
## lm(formula = SQRTPPOV ~ PFHH + PUNEM + PBLK + P65UP + METRO +
##     PHSPLUS, data = south.sf)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.234660 -0.032609 -0.000461  0.029598  0.249979
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.34195    0.01074  31.829  < 2e-16 ***
## PFHH         0.64126    0.04349  14.747  < 2e-16 ***
## PUNEM        1.58472    0.06938  22.841  < 2e-16 ***
## PBLK        -0.10640    0.01647  -6.459 1.46e-10 ***
## P65UP        0.17417    0.04106   4.242 2.37e-05 ***
## METRO       -0.02633    0.00338  -7.790 1.31e-14 ***
## PHSPLUS     -0.30564    0.01495 -20.439  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05042 on 1380 degrees of freedom
## Multiple R-squared:  0.7286, Adjusted R-squared:  0.7275
## F-statistic: 617.6 on 6 and 1380 DF,  p-value: < 2.2e-16
```

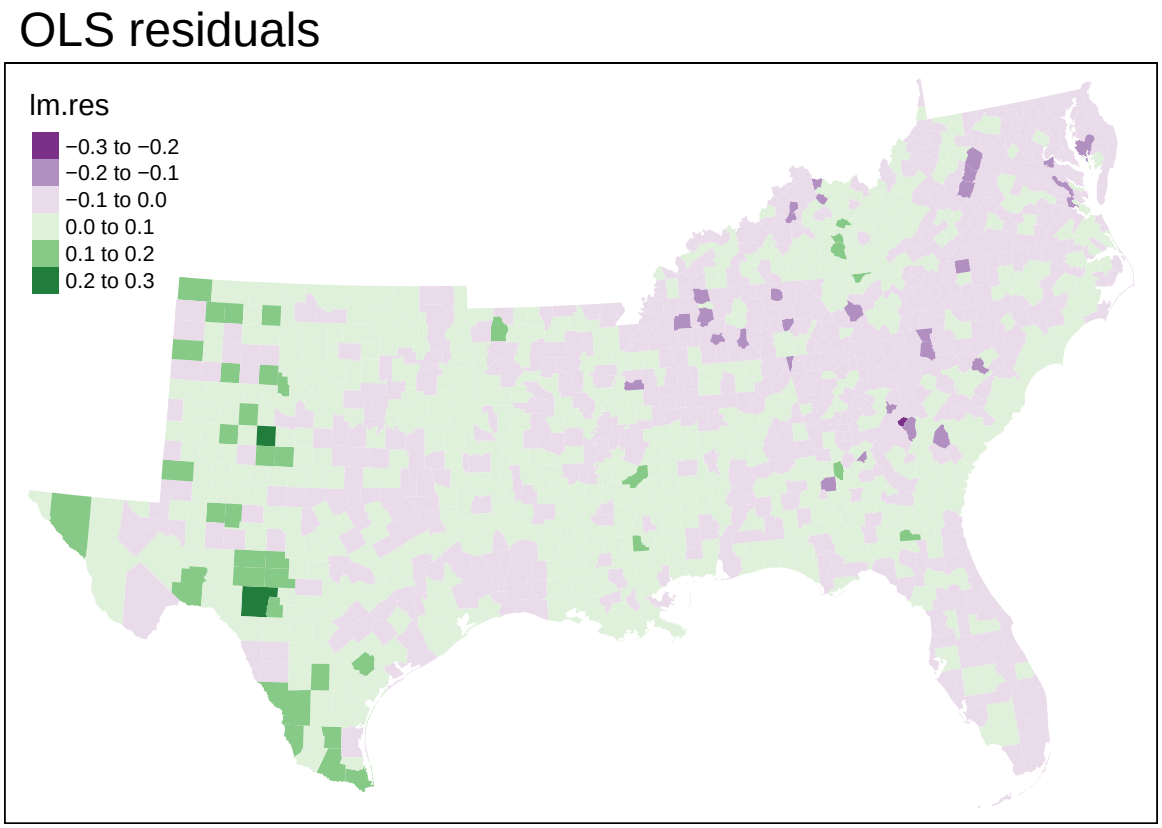
```
moran.test(residuals(fit1), south_listw)
```

```
##
##  Moran I test under randomisation
##
## data:  residuals(fit1)
## weights: south_listw
##
## Moran I statistic standard deviate = 22.706, p-value < 2.2e-16
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##      0.3657562577     -0.0007215007      0.0002604946
```

A continuación podemos cartografiar los residuos del modelo, que muestran un patrón espacial claro:

```
south.sf$lm.res <- residuals(fit1)

tm_shape(south.sf) +
  tm_fill("lm.res", palette = "PRGn") +
  tm_layout(main.title = "OLS residuals")
```



La construcción del modelo GWR usa el paquete **spgwr** y requiere dos pasos, el primero para evaluar el mejor tamaño de la ventana y el segundo para construir y diagnosticar los modelos locales elaborados a partir de esta venta. La ventana puete tener:

- Tamaño fijo: cada ventana tendrá el mismo ancho de banda (tamaño), por lo que los modelos en áreas con pocos datos tendrán menos observaciones.
- Adaptable: en lugar de establecer un tamaño de ventana único, la ventana de cada modelo se elige para capturar el mismo número de observaciones.

En nuestro caso, usaremos el segundo método, configurando el argumento `adapt = TRUE` en la función GWR. ¿Por qué? La elección está dictada por la gran variabilidad en el tamaño de los condados, por lo que las áreas con condados más pequeños tendrán un conjunto de observaciones mucho más denso. Primero necesitamos extraer los centroides de cada condado (polígono) para usarlos en los cálculos de distancia (estos se usarán para seleccionar las ubicaciones dentro de una ventana alrededor de un punto de interés).

```
coords <- st_coordinates(st_centroid(south.sf))
```

Empiece por encontrar el valor de `q`: la proporción de puntos en cada ventana:

```
south.bw <- gwr.sel(SQRTPPOV ~ PFHH + PUNEM + PBLK + P65UP + METRO + PHSPLUS,
                    data = south.sf,
                    coords = coords,
                    adapt = TRUE,
                    gweight = gwr.Gauss,
                    method = "cv",
                    verbose = TRUE)
```

```
## Adaptive q: 0.381966 CV score: 3.066879
## Adaptive q: 0.618034 CV score: 3.221821
## Adaptive q: 0.236068 CV score: 2.927806
## Adaptive q: 0.145898 CV score: 2.798224
## Adaptive q: 0.09016994 CV score: 2.67092
## Adaptive q: 0.05572809 CV score: 2.550302
## Adaptive q: 0.03444185 CV score: 2.447888
## Adaptive q: 0.02128624 CV score: 2.389188
## Adaptive q: 0.01315562 CV score: 2.383642
## Adaptive q: 0.0153006 CV score: 2.375256
## Adaptive q: 0.01677638 CV score: 2.373892
## Adaptive q: 0.01659908 CV score: 2.374078
## Adaptive q: 0.01849899 CV score: 2.376264
## Adaptive q: 0.01743436 CV score: 2.374001
## Adaptive q: 0.01704878 CV score: 2.373764
## Adaptive q: 0.01708947 CV score: 2.373761
## Adaptive q: 0.01713016 CV score: 2.373761
## Adaptive q: 0.01708947 CV score: 2.373761
```

```
south.bw
```

```
## [1] 0.01708947
```

El paquete incluye varias opciones (argumentos):

- `adapt = TRUE`: La función usará un ancho de banda adaptativo para asegurar que se usa aproximadamente el mismo número de observaciones en cada ventana local para construir un modelo.
- `gweight = gwr.Gauss`: Usa pesos gaussianos para ajustar cada modelo (los puntos más cercanos tendrán un mayor peso en el ajuste del modelo de regresión -OLS-).
- `method ="cv"`: Usa un procedimiento de validación cruzada para probar diferentes tamaños de ancho de banda/número de puntos.

Podemos estimar el número aproximado de observaciones en cada ventana de la siguiente manera:

```
dim(south.sf)[1] * south.bw
```

```
## [1] 23.7031
```

La salida de esta función enumera el valor cambiante de `q`, así como el RMSE (raíz del error cuadrático medio; medida de uso frecuente obtenida a partir de las diferencias entre los valores predichos por un modelo y los valores observados) de la validación cruzada. El valor de `q` varía hasta que no se produzcan mejoras en el RMSE. Tenga en cuenta que usamos la salida del paso anterior para establecer el tamaño de la ventana (`adapt = south.bw`), y establecemos la forma de la ventana como gaussiana (`gweight = gwr.Gauss`).

Por último, elaboramos el conjunto final de modelos usando la función `gwr ()`.

```
south.gwr <- gwr(SQRTPPOV ~ PFHH + PUNEM + PBLK + P65UP + METRO + PHSPLUS,
                 data = south.sf,
                 coords = coords,
                 adapt = south.bw,
                 gweight = gwr.Gauss,
                 hatmatrix = TRUE)

south.gwr
```

```
## Call:
## gwr(formula = SQRTPPOV ~ PFHH + PUNEM + PBLK + P65UP + METRO +
##     PHSPLUS, data = south.sf, coords = coords, gweight = gwr.Gauss,
##     adapt = south.bw, hatmatrix = TRUE)
## Kernel function: gwr.Gauss
## Adaptive quantile: 0.01708947 (about 23 of 1387 data points)
## Summary of GWR coefficient estimates at data points:
##                     Min.    1st Qu.     Median    3rd Qu.       Max.  Global
## X.Intercept.  0.2005339  0.2799681  0.3398143  0.4037562  0.5059632  0.3420
## PFHH         -0.0405331  0.5569098  0.7030943  0.8323044  1.2702785  0.6413
## PUNEM         0.1308578  0.7678957  1.0349647  1.4880431  2.2959962  1.5847
## PBLK         -0.5843126 -0.1491605 -0.0610091  0.0328856  0.5007957 -0.1064
## P65UP        -1.0307935  0.0651806  0.2045079  0.3702337  0.9033343  0.1742
## METRO        -0.0527382 -0.0229680 -0.0144405 -0.0084449  0.0096279 -0.0263
## PHSPLUS      -0.5564329 -0.3728045 -0.3012876 -0.2458864 -0.1088166 -0.3056
## Number of data points: 1387
## Effective number of parameters (residual: 2traceS - traceS'S): 244.3255
## Effective degrees of freedom (residual: 2traceS - traceS'S): 1142.675
## Sigma (residual: 2traceS - traceS'S): 0.03911912
## Effective number of parameters (model: traceS): 173.8689
## Effective degrees of freedom (model: traceS): 1213.131
## Sigma (model: traceS): 0.03796614
## Sigma (ML): 0.03550685
## AICc (GWR p. 61, eq 2.33; p. 96, eq. 4.21): -4923.036
## AIC (GWR p. 96, eq. 4.22): -5149.69
## Residual sum of squares: 1.748641
## Quasi-global R2: 0.8647189
```

La salida del modelo proporciona estadísticas resumidas sobre el rango de coeficientes en todos los modelos construidos ($n = 1387$, el número de condados). En los diagnósticos del modelo, encontrará un valor AIC, que puede usarse en la comparación de modelos, y un valor de R^2, que debe tratarse con cierta precaución.

La salida de la función `gwr ()` incluye un objeto (lista) de gran tamaño y con mucha información en él. Un objeto de esta lista es un "SpatialPointsDataFrame". Esta es la forma más antigua de objeto espacial en R y contiene información diversa sobre los modelos locales. Ahora usaremos esto para trazar algunos de los resultados del modelo. Es posible convertirlos en un objeto `sf`, pero es más fácil simplemente asignar nuevas columnas en el objeto `south.sf` existente con los resultados que queremos visualizar.

```
class(south.gwr$SDF)
```

```
## [1] "SpatialPointsDataFrame"
## attr(,"package")
## [1] "sp"
```

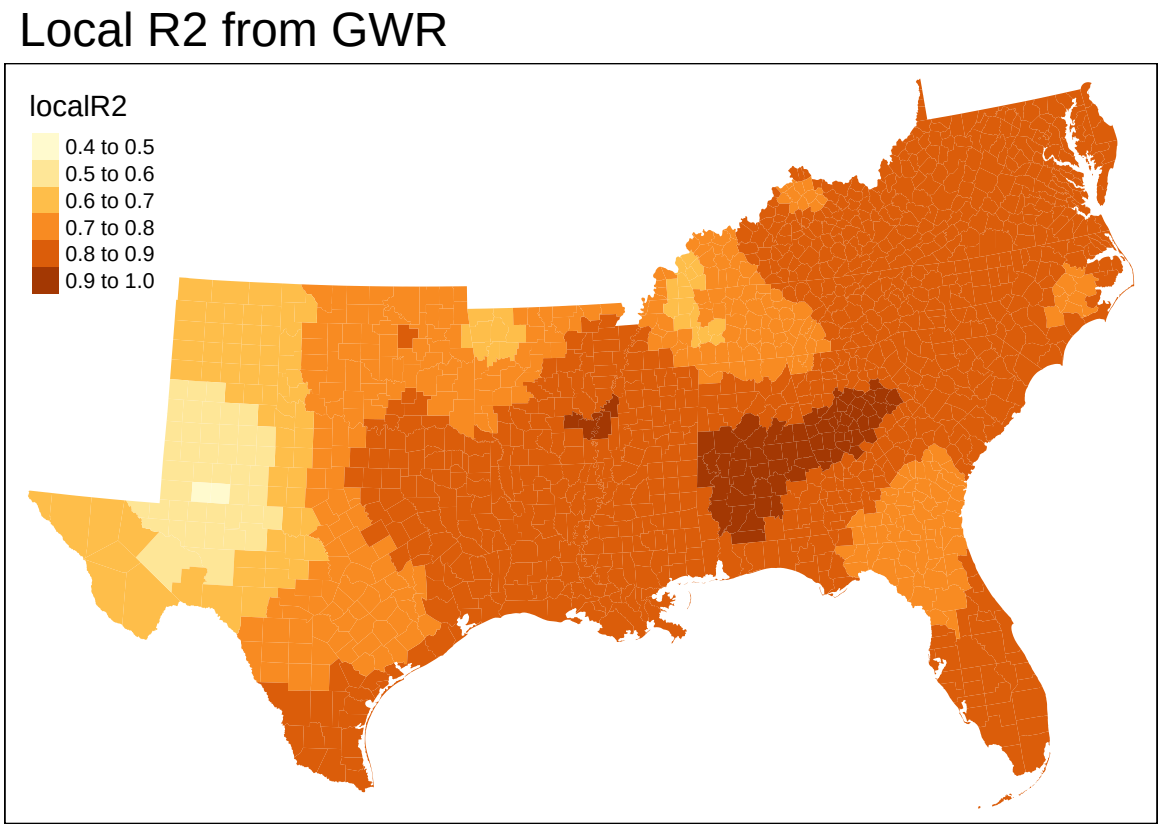
Se puede acceder a los resultados individuales desde el objeto `SpatialPointsDataFrame` usando la siguiente notación:

```
south.gwr$SDF$localR2
```

Primero cartografiamos los valores de R^2 para cada medida local. Las áreas con valores bajos pueden indicar una especificación incorrecta del modelo (por ejemplo, faltan variables independientes, caso del oeste de Texas).

```
south.sf$localR2 <- south.gwr$SDF$localR2

tm_shape(south.sf) +
  tm_fill("localR2") +
  tm_layout(main.title = "Local R2 from GWR")
```



Y finalmente graficamos los coeficientes para una de las variables independientes, el porcentaje de la población de 65 años o más (`P65UP`).

```r
south.sf$beta_P65UP <- south.gwr$SDF$P65UP
tm_shape(south.sf) + tm_fill("beta_P65UP", palette = "PRGn", n = 9) +
  tm_layout(main.title = "Coefficient for P65UP")
```