Modelos de predicción de precios para datos georreferenciados

Maximiliano S. Lioi

2023-03-03

Introducción

El objetivo de predecir el precio de las viviendas en una determinada ubicación utilizando datos georreferenciados trae consigo una serie de desafíos que debemos enfrentar, tales como: la autocorrelación espacial y la heterogeneidad espacial sobre las muestras; estas propiedades afectan suposiciones clásicas para asegurar la calidad de la predicción en modelos más simples de regresión, como puede ser la identica e independiente distribución de los datos o la homocedasticidad de estos.

El presente documento se enfoca en el desarrollo de modelos de regresión espacial, los cuales adaptan la naturaleza espacial de los datos en sus formulaciones, tomar estas cualidades en cuenta permite obtener modelos más precisos, estos resultados pueden ser útiles para diversos ámbitos de la industria, en cuyas tomas de decisiones intervengan datos que están asociados a una componente geográfica, como es el caso de la predicción de precios de vivienda dentro del mercado inmobiliario.

En el contexto de datos espaciales [5], trabajamos con un conjunto de observaciones de datos asociados con una componente espacial $\{(x(s_i), y(s_i))|\ i \in \mathbb{N}, 1 \leq i \leq n\}$ donde $n \in \mathbb{N}$ es la cantidad de muestras, $s_i \in \mathbb{R}^2$ es un vector de coordenadas espaciales, $x(s_i) \in \mathbb{R}^m$ son las variables explicativas del modelo e $y(s_i) \in \mathbb{R}$ es la respuesta a dichas variables.

Este mismo conjunto también puede ser descrito en formato matricial como $(X, Y) \in \mathbb{R}^{(m+1)\times n}$ donde $X = [x(s_1), x(s_2), \ldots, x(s_n)]^T \in \mathbb{R}^{m\times n}$ e $Y = [y(s_1), y(s_2), \ldots, y(s_n)]^T \in \mathbb{R}^n$.

El problema de la predicción consiste en que, dado un conjunto de muestras de data espacial, buscamos modelar una función f tal que $Y = f(X) + \varepsilon$ donde ε es un término para el error, una vez que el modelo se entrena, minimizando el error, puede usarse para precedir la respuesta en otra locación dada sus variables explicativas.

La predicción espacial se diferencia de la predicción usual, puesto que en este último, se asume que las muestras son independientes e identicamente distribuidas (i.i.d) y por lo tanto, dado un modelo entrenado en la función f, podemos usarlo para predecir $y(s) = f(x(s))$ para todo s, sin embargo, la suposición de que los datos son i.i.d no se cumple al trabajar con datos espaciales, esto debido a la relación implícita que existe entre posiciones cercanas en una región.

Acorde a la primera ley de la geografía de Tobler W.R: "Todo se relaciona con todo lo demás, pero las cosas más cercanas se relacionan más que las cosas distantes" [7], este fenómeno se conoce como autocorrelación espacial, e ignorarlo puede afectar la precisión de modelos que tengan el supuesto de una muestra i.i.d, pues perdemos esta propiedad sobre los datos.

Respecto de los desafíos presentes asociados a la predicción espacial, abordamos los siguientes a lo largo del documento:

- *Autocorrelación espacial* → como indica Tobler con su primera ley de geografía, los datos espaciales no son estadísticamente independientes entre sí, al contrario estos, están correlacionados, y la intensidad de dicha relación depende de la distancia que exista entre estos, siendo mayor cuando los datos están cerca entre sí, dicho fenómeno interviene cuando usamos modelos de predicción que toman por hipótesis una distribución independiente e identica entre los datos, como es el caso de la regresión lineal.
- *Heterogeneidad espacial* → cuando fragmentamos el espacio para el análisis de los datos y le asignamos a cada sector algún atributo, los datos que conforman cada zona no siguen una distribución idéntica, modelos clásicos que no toman en cuenta este fenómeno, por medio de la función que represente al proceso, los valores que representan la relación entre los regresores y la variable dependiente se asumen constantes para los regresores independiente de su posición geográfica, esto se conoce como estacionariedad. Dichas suposiciones permiten simplificar la creación de modelos, sin embargo, cuando no se cumple este principio, como es típico con datos demográficos, puede interferir en la precisión y es por ello que necesitamos técnicas que tomen en cuenta las variaciones locales.

Para el análisis de datos haremos uso de *tidycensus*, paquete de R diseñado para facilitar el proceso de adquisición y manejo de datos sobre la población, proporcionados por la oficina del censo de EE.UU, acompañados de otros paquetes para el uso de modelos de predicción, visualización, entre otros.

En *Analyzing US Census Data - Kyle Walker* se muestra como hacer uso de las librerías, entre ellas:

- tidycensus : Paquete de R diseñado para facilitar procesos de adquirir y trabajar data de US Census, busca distribuir los datos del censo en un formado compatible con tidyverse, además busca agilizar el proceso del tratado de datos para aquel que esté trabajando en el analisis de datos. Ch2.
- *tidyverse* : Coleción de paquetes de R diseñados para la ciencia de datos tales como *ggplot2* para la visualización de data, *readr* para importar y exportar bases de datos, *tidyr* para la remodelación de datos, entre otros. Ch3.
- *tigris* : Paquete de R que busca simplificar procesos para los usuarios de obtención de información y uso de data con atributos geográficos (data espacial, Census geographic dataset), data tipo *sf* (simple features) viene con atributos de geometría (vector data type, tipicamente representados por puntos lineas o poligonos). Ch5.
- *ggplot2* : Paquete de R enfocado en la visualización de data, nos permite realizar mapas con información de US Census data. Ch6.
- *spatialreg & GWmodel* : Paquete diseñado para la aplicación de modelos de regresión espacial

Importando datos

Se importan datos con *tidycensus* provenientes de la American Community Survey (ACS) (2016-2020), haciendo uso de la función get_acs()

Tomamos datos con nivel geográfico de distritos para la ciudad de NY, para ello pedimos datos de los condados de: 'Bronx', 'Kings', 'New York', 'Queens', 'Richmond'

```
# Obtiene datos de la ACS y transforma a tipo NYC EPSG
nyc_data <- get_acs(
  geography = "tract",
  variables = variables_to_get,
  state = "NY",
  county = nyc_counties,
  geometry = TRUE,
  output = "wide",
  year = 2020,
  key = Sys.getenv("CENSUS_API")) %>%
  st_transform(2263)
```

Las variables que importamos, para usar en los distintos modelos de predicción, son las siguientes:

- **median_value** : El valor medio de la vivienda del tramo censal, nuestra variable a predecir
- median_rooms : Cantidad media de habitaciones por casa en el tramo censal
- total_population : Población total en el tramo censal
- median_age : Edad media de la población en el tramo censal
- median_year_built : Año promedio donde se construyó la vivienda
- median_income : Ingreso medio de los hogares en el tramo censal
- pct_college : Porcentaje de la población de 25 años o más con un título universitario de cuatro años
- pct_foreign_born: Porcentaje de la población que nació fuera de EE.UU
- pct_white : Porcentaje de la población que se identifica como blanco no-hispano, se sigue la misma lógica con pct_black, pct_asian, pct_hispanic.
- percent_ooh : Porcentaje de viviendas en el tramo censal que están siendo ocupadas por sus propietarios.

Los detalles para descargar información de las distintas bases de datos que proporciona *tidycensus* y las distintas variables que tenemos a disposición se encuentran en el **Anexo 3**.

Más detalles sobre la estructura del código se encuentran en la sección 8.2.1 de *Analyzing US Census Data*, en el libro se toma la misma variable "median home value", pero a diferencia de nuestro caso, se toman los tramos censales en Dallas-Fort Worth metropolitan area, en consecuencia también cambian el código usando en st_transform, a un sistema de referencias apropiado para North Texas (code 32148), a diferencia del caso NYC donde usamos code 2263

Preparando la data para modelar

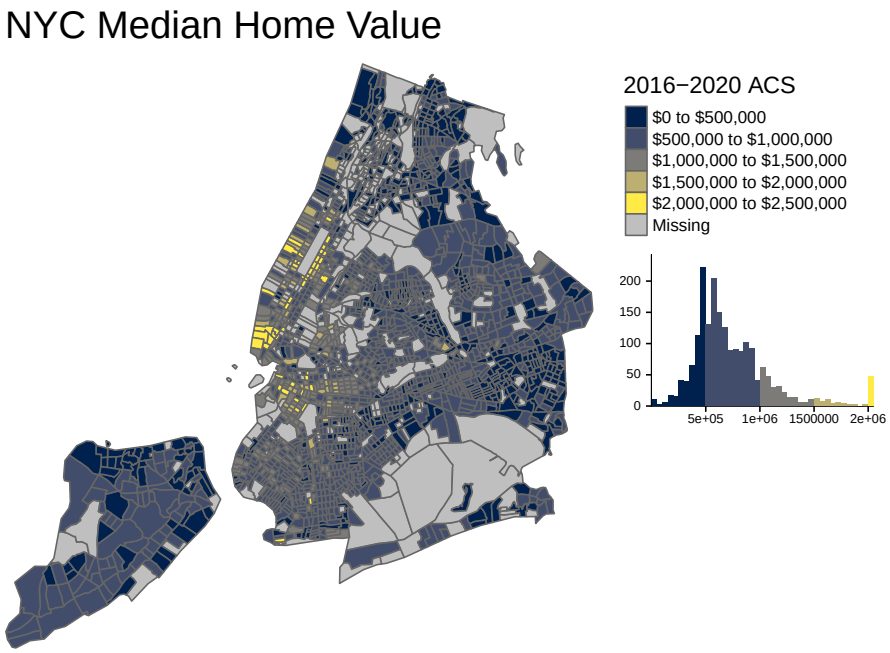
Se realiza una limpieza de las muestras (data scrubbing), identificando datos incompletos, incorrectos o inexactos.

Visualizamos un resumen de los datos, mostrando sus estadísticos principales, tales como mínimo, máximo, promedio, mediana, cuartiles y valores perdidos.

median_valueE <chr>	median_valueM <chr>	median_roomsE <chr>	median_roomsM <chr>	total_populationE <chr>	total_populationM <chr>	median_ageE <chr>
Min. : 13700	Min. : 2462	Min. : 1.400	Min. :0.1000	Min. : 0	Min. : 3.0	Min. :11.60
1st Qu.: 491700	1st Qu.: 50119	1st Qu.: 3.600	1st Qu.:0.3000	1st Qu.: 2192	1st Qu.: 389.0	1st Qu.:33.60
Median : 640100	Median : 86720	Median : 4.100	Median :0.3000	Median : 3363	Median : 567.0	Median :37.40
Mean : 731462	Mean : 144148	Mean : 4.285	Mean :0.3579	Mean : 3601	Mean : 608.8	Mean :38.17
3rd Qu.: 885100	3rd Qu.: 169755	3rd Qu.: 4.900	3rd Qu.:0.4000	3rd Qu.: 4684	3rd Qu.: 779.5	3rd Qu.:42.00
Max. :2000001	Max. :1343387	Max. :10.000	Max. :3.0000	Max. :16600	Max. :2467.0	Max. :85.10
NA's :350	NA's :398	NA's :109	NA's :110	*NA*	*NA*	NA's :92

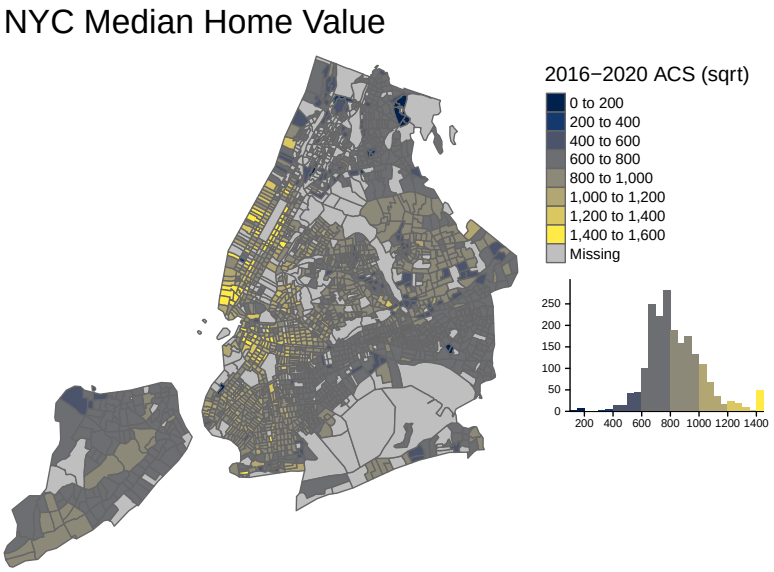
La recolección de datos incluye para cada variable otra columna asociada que representa el margen de error de los datos (aquellas que terminan en 'M'), sin embargo, no haremos uso de ello y por tanto la eliminamos del modelo.

Nuestra variable dependiente para los modelos de regresión será el valor promedio de la vivienda en NYC (median_value).



El histograma presenta una asimetría a la derecha, i.e, existe una población no menor cuyas viviendas poseen un valor muy alejado de la mediana, usualmente es más facil trabajar con la variable dependiente distribuida normal, por sus buenas propiedades y teoremas asociados, además que en la práctica, permite resolver problemas como la normalidad de los residuos, para ello podemos aplicar una transformación raíz cuadrada, por otro lado tenemos datos sin información, para esto aplicamos herramientas de R para el filtrado.

La distribución de la variable dependiente aplicada esta transformación se ve de la siguiente manera



Eliminamos aquellas variables con data NA, y aquellas columnas con información del margen de error, por lo demás, como tenemos la cantidad de población, y el área que cubre cada distrito (polygon), creamos la variable pop_density, esto pues buscamos transformar la información de la población por distrito de manera que represente mejor la relación que tiene con la variable de salida, en este caso, el valor medio de la vivienda

- pop_density → mide densidad de población en el tramo censal por metros cuadrados

```
# Preparando la data
nyc_data_prepped <- nyc_data %>%
  mutate(pop_density = as.numeric(set_units(total_populationE / st_area(.),
 "1/km2"))) %>%
  select(!ends_with("M")) %>% # Eliminamos columnas de margen de error
  rename_with(.fn = ~str_remove(.x, "E$")) %>%
  na.omit() # Elimina valores NA
```

Para ver como visualizar los datos haciendo uso de la geometría incorporada por *tidycensus*, ver **Anexo 4** en donde se hace una introducción de los paquetes y la sintaxis.

Autocorrelación espacial: Efectos sobre una regresión lineal

Existen múltiples modelos estadísticos que se rigen por hipótesis que requieren de una distribución i.i.d de los datos, sin embargo, como mencionamos anteriormente, esta cualidad se pierde al trabajar con datos espaciales.

En esta sección, se presenta un estadístico para medir la autocorrelación espacial presente en los datos, interpretaciones de dicho estadístico y como este fenómeno afecta la predicción de modelos, cuyas hipótesis se ven comprometidas por la naturaleza de la muestra, como es el caso de la regresión lineal

Índice de Moran

Para medir la autocorrelación espacial presente en los datos, hacemos uso del índice de Moran I [5] [6], este coeficiente nos entrega una magnitud de la intensidad de este fenómeno para n observaciones de una misma variable, donde las muestras poseen una componente espacial.

Definimos el índice de Moran como

$$I = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(y(s_i)-\bar{y})(y(s_j)-\bar{y})}{(\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij})\sum_{i=1}^{n}(y(s_i)-\bar{y})^2/n}$$

O de manera equivalente

$$= \frac{n}{W}\frac{\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(y(s_i)-\bar{y})(y(s_j)-\bar{y})}{\sum_{i=1}^{n}(y(s_i)-\bar{y})^2}$$

donde $\bar{y} = \sum_{i=1}^{n} y(s_i)/n$ con n el número total de muestras, a partir de ahora, para simplificar la notación, $y(s_i) = y_i$ dejando implicita la componente espacial, además $W = \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}$, y de forma abreviada, $W = (w_{ij})_{i,j=1}^{n}$, que consideramos como una **matriz de pesos estandarizada**.

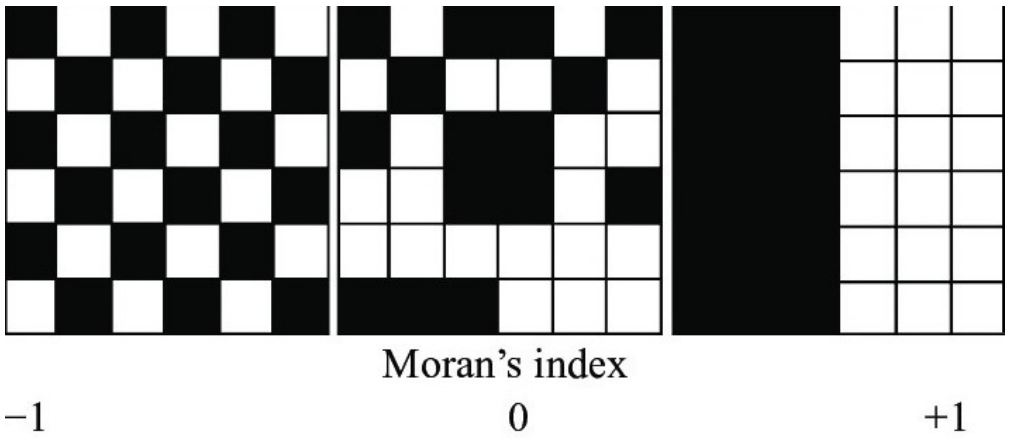
Entenderemos por **matriz de pesos estandarizada** aquella que cumple las siguientes propiedades

- Supondremos $w_{ij} \geq 0 \ \ \forall i, j \in \{1, \ldots, n\}$, pues w_ij es una magnitud de la influencia que hay entre los vecinos i y j
- Típicamente, $w_{ii} = 0 \ \ \forall i \in \{1, \ldots, n\}$, pues el hecho de que un punto tenga influencia con el mismo puede generar ruido en el modelo
- Suponemos que la influencia que tiene y_i sobre y_j es la misma que la de y_j sobre y_i, esto es, $w_{ij} = w_{ji} \ \ \forall i, j \in \{1, \ldots, n\}$, es decir, W es simétrica
- *Row-normalized*: Para cada $i \in \{1, \ldots, n\}$ se tiene que $\sum_{j=1}^{n} w_{ij} = \alpha$ para algún $\alpha \in \mathbb{R}$ esto pues, entendemos la suma $\sum_{j=1}^{n} w_{ij}$ como la influencia total que existe entre el vecino i y todos sus vecinos j, por lo que pedimos que la influencia total sea la misma para cada vecino $i \in \{1, \ldots, n\}$, de manera que el nivel de influencia esté estandarizado para cada vecino, típicamente tomamos $\alpha = 1$.

El índice de Moran, toma valores entre -1 y 1 dado un muestreo aleatorio $(y(s_i))_{i=1}^{n}$, para una demostración detallada de esta propiedad, ver **Anexo 5**.

Este estadístico mide que tan similar es una variable respecto de las variables cercanas, por ejemplo, si este se acerca a -1 nos dice que cada en cada región, esta tiende a tener valores distinto de su entorno, y en caso contrario, si se acerca a 1, cada región es similar a su entorno, resumiendo

- $I \approx -1 \rightarrow$ Se clusterizan valores distintos entre sí (dispersión perfecta)
- $I \approx 0 \rightarrow$ No existe una clara relación entre los valores de los datos y sus entornos, es decir, no hay pruebas de autocorrelación espacial.
- $I \approx 1 \rightarrow$ Se clusterizan valores similares entre sí



Comentando sobre el significado de este valor, al igual que la correlación usual, en donde vemos la relación entre 2 variables, tomando una como variable de entrada y la otra su variable de respuesta, la autocorrelación es similar, pero la variable de respuesta es la misma variable de entrada, tomando en cuenta el factor espacial por medio de nuestra matriz de pesos.

Dada su estructura, vemos que el índice de Moran es una auto-covarianza espacial estandarizada [2].

Además, si consideramos la hipótesis nula:

$$H_0 : \text{No existe autocorrelación espacial}$$

La cual consiste en suponer que, dada una permutación de la muestra, es decir, cambiando el orden en el que vienen los datos $(x_i)_{i=1}^N$, y fijando la matriz de pesos antes del reordenamiento, no se debiese tener un impacto sobre el valor de I, pues estos no muestran relacionarse entre sí, se tiene que

$$\mathbb{E}(I) = -\frac{1}{N-1}$$

Esto es consistente con el modelo, pues para una muestra grande, de no existir autocorrelación espacial, esperamos que $I \approx 0$

Luego, podemos usar esta propiedad para inferencias estadísticas por medio del p-valor

Para una demostración detallada de este hecho, ver **Anexo 5**.

Regresión Lineal

Planteamos un modelo de regresión lineal para los datos importados, dicho modelo no toma en cuenta las componentes espaciales y nuestro objetivo es observar el efecto que esto tiene en la calidad de la precisión

La formulación es la siguiente:

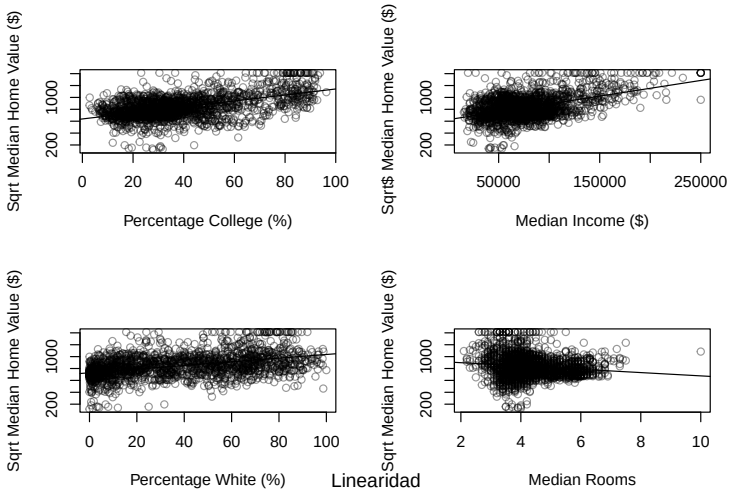
$$\sqrt{median\ value} = \alpha + \beta_1(median\ rooms) + \beta_2(median\ income) + \beta_3(pct\ college)$$
$$+\beta_4(pct\ foreign\ born) + \beta_5(pct\ white) + \beta_6(median\ age)$$
$$+\beta_7(percent\ ooh) + \beta_8(pop\ density) + \beta_9(total\ population) + \varepsilon$$

De donde, a partir de n observaciones de la variable dependiente junto a sus regresores, buscamos estimar los valores α y β_i, mediante mínimos cuadrados, es decir, tenemos para cada observación un error asociado ε_i, y minimizamos $\sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n}(y_i - \alpha - \sum_{j=1}^{m} \beta_j x_{ij})^2$ donde y_i es la observación de la variable independiente, y x_{ij} los m regresores asociados a la observación i.

Un modelo de regresión lineal requiere de una serie de hipótesis para asegurar la calidad de las estimaciones de β (supuestos de Gauss-Márkov) tales como:

- **Linearidad**: Los regresores poseen una relación lineal con la variable de respuesta
- **Independencia**: Las observaciones representan una muestra aleatoria distribuida identica e independiente, de manera que es generalizable para el total de la población
- **Error con media cero**: Consideramos el error de cada observación ε_i variable aleatoria tal que $\mathbb{E}(\varepsilon_i) = 0$
- **Normalidad**: Suponemos que ε_i sigue una distribución normal de la forma $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$
- **Homocedasticidad e incorrelación**: Los errores de las observaciones del modelo poseen la misma varianza σ^2 y son tales que $Cov(\varepsilon_i, \varepsilon_j)$.

Realizando un análisis exploratorio visualizamos una tendencia lineal en los datos entre los regresores y $\sqrt{median\ value}$



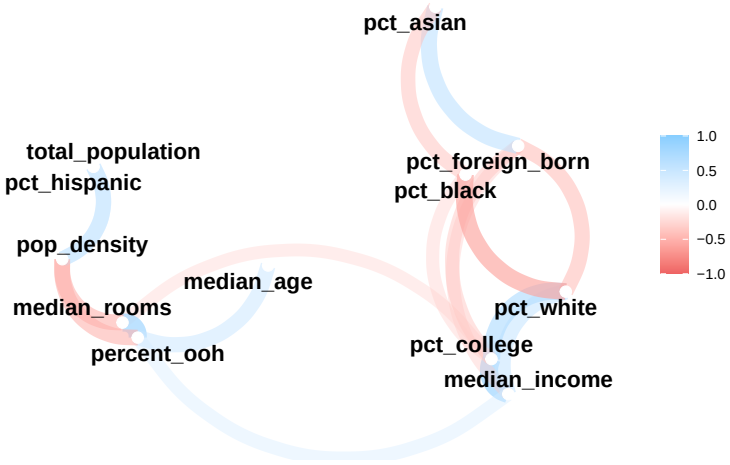
Además, para el análisis de datos, el paquete *corrr*, nos permite calcular la matriz de correlación asociada [8]

```
library(corrr)

nyc_estimates <- nyc_data_prepped %>%
  select(-GEOID, -median_value ) %>%
  st_drop_geometry()

correlations <- correlate(nyc_estimates, method = "pearson")

correlations[is.na(correlations)] <- 0
```

	term	median_rooms	total_population	median_age	median_income	pct_college	pct_foreign_born
1	median_rooms	0.000000e+00	-2.755315e-01	1.862939e-01	8.841287e-02	-3.042464e-01	-3.492071e-02
2	total_population	-2.755315e-01	0.000000e+00	-1.053697e-01	-5.793610e-05	1.179050e-01	-1.067049e-01
3	median_age	1.862939e-01	-1.053697e-01	0.000000e+00	1.794448e-01	1.401042e-01	1.544935e-01
4	median_income	8.841287e-02	-5.793610e-05	1.794448e-01	0.000000e+00	7.502369e-01	-3.434698e-01
5	pct_college	-3.042464e-01	1.179050e-01	1.401042e-01	7.502369e-01	0.000000e+00	-3.935039e-01
6	pct_foreign_born	-3.492071e-02	-1.067049e-01	1.544935e-01	-3.434698e-01	-3.935039e-01	0.000000e+00
7	pct_white	-2.093989e-02	5.620708e-02	1.239277e-01	5.181632e-01	6.518286e-01	-4.723701e-01
8	pct_black	1.526903e-01	-1.021483e-01	-4.425499e-02	-2.146035e-01	-3.249169e-01	3.500025e-03
9	pct_hispanic	-2.755315e-01	1.000000e+00	-1.053697e-01	-5.793610e-05	1.179050e-01	-1.067049e-01
10	pct_asian	-3.019054e-02	-4.319723e-02	1.686201e-01	-4.314301e-02	-1.434416e-02	5.649252e-01
11	percent_ooh	7.479068e-01	-2.773636e-01	4.775084e-01	3.640753e-01	3.751176e-02	-3.817959e-02
12	pop_density	-5.257080e-01	4.827097e-01	-2.284217e-01	-6.815065e-02	1.562171e-01	6.102068e-04
13	residuals	-7.277834e-17	-6.226377e-17	-3.538226e-17	-2.886366e-17	-3.196077e-17	-4.313835e-17
14	lagged_residuals	-4.075953e-02	-3.706601e-02	-9.610488e-02	1.234156e-01	9.782962e-02	-1.749310e-02

Y podemos visualizar esta información haciendo uso de network_plot()

```
network_plot(correlations)
```



En donde vemos el tipo de correlación que tienen los regresores entre sí.

Se puede realizar un análisis de componentes principales (PCA) con el fin de disminuir la dimensionalidad del problema, en donde reducimos la cantidad de regresores tomando combinaciones lineales de estos, ver *sección 8.2.5: 'Dimension reduction with principal components analysis'* de *Analyzing US Census Data*

Para la calibración de los parámetros β y α del modelo de regresión lineal, podemos usar la función *lm* que viene incluida con los paquetes básicos del lenguaje R.

```
# Fórmula entre variables dependientes y regresores
formula <- "sqrt(median_value) ~ median_rooms + median_income +
pct_college + pct_foreign_born + pct_white + pct_black + pct_hispanic +
pct_asian + median_age + percent_ooh + pop_density"

# Regresión lineal mediante lm
model1 <- lm(formula = formula, data = nyc_data_prepped)
summary(model1)
```

```
Call:
lm(formula = formula, data = nyc_data_prepped)

Residuals:
    Min      1Q  Median      3Q     Max
-657.75  -78.17    0.32   77.44  683.88

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)      5.048e+02  3.974e+01  12.704  < 2e-16 ***
median_rooms     6.209e+01  6.465e+00   9.605  < 2e-16 ***
median_income    2.091e-03  1.704e-04  12.272  < 2e-16 ***
pct_college      1.696e+00  3.624e-01   4.680 3.06e-06 ***
pct_foreign_born -6.281e-01  3.245e-01  -1.936 0.053039 .
pct_white        2.421e+00  2.376e-01  10.192  < 2e-16 ***
pct_black        1.003e+00  2.074e-01   4.838 1.42e-06 ***
pct_hispanic    -1.399e-02  2.127e-03  -6.577 6.16e-11 ***
pct_asian        3.108e+00  3.000e-01  10.360  < 2e-16 ***
median_age      -2.166e+00  5.984e-01  -3.620 0.000302 ***
percent_ooh     -4.513e+00  2.807e-01 -16.077  < 2e-16 ***
pop_density      1.046e-03  3.401e-04   3.075 0.002133 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 146.8 on 1957 degrees of freedom
Multiple R-squared:  0.4569,    Adjusted R-squared:  0.4539
F-statistic: 149.7 on 11 and 1957 DF,  p-value: < 2.2e-16
```

Haciendo uso de esta función, obtenemos el valor de los coeficientes buscados, el estimado de la variable 'Intercept' se corresponde al valor de α mientras que el estimado del resto de variables se corresponde a los $(\beta_i)_{i=1}^m$ asociados.

La columna '$P(> |t|)$' corresponde al **p-valor**, un p-valor alto nos dice que la variable no es significativa en el resultado, típicamente pedimos que sea menor a 0.05 para que la variable se considere significativa, los grados de significancia para el modelo vienen jerarquizados como dice 'Signif. codes', es decir, cuando el p-valor es menor a 0.001, consideramos que el regresor es altamente significativo en el modelo, menos significativo si el p-valor esta entre 0.001 y 0.01 y así sucesivamente.

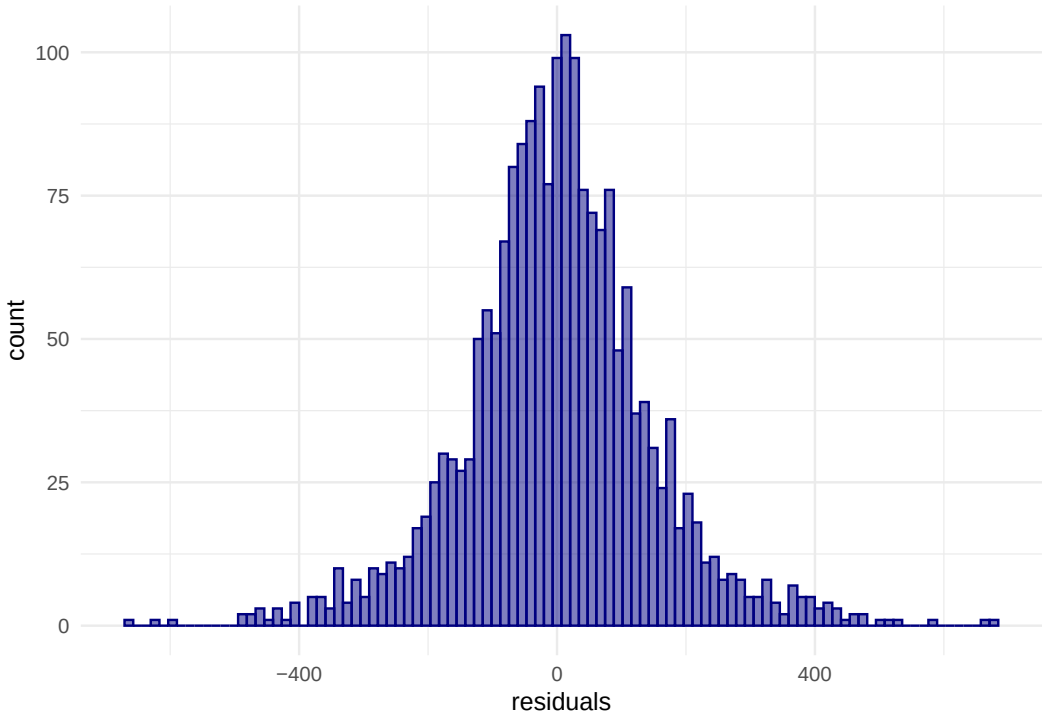
Aquellas variables con mayor p-value son pct_foreign_born, median_age, pop_density, notamos tambien que las primeras dos variables se correlacionan negativamente con median_value, es decir, a mayor cantidad de nacidos extranjeros y edad promedio, se tiene que el valor medio de la vivienda disminuye. Al contrario, si aumenta la densidad poblacional, tenemos que el valor medio de la vivienta aumenta.

Observando el valor R^2, coeficiente que mide el porcentaje de varianza de la variable dependiente que es explicado con las variables del modelo, un $R^2 = 0.4569$ nos revela que **el modelo es deficiente para la predicción**, una de las razones de esto es debido a la autocorrelación espacial presente en los datos.

Para hacer cuenta de esto último, se estudian los residuos del modelo, que se corresponden a la diferencia entre el valor observado y el valor predicho por el modelo ya entrenado, para ello hacemos uso de la función residuals().

```
#Añadimos los residuos a la data que estamos trabajando
nyc_data_prepped$residuals <- residuals(model1)
```

Mediante un histograma vemos la distribución que siguen los residuos



Vemos que los residuos siguen aparentemente una distribución normal, como se mencionó antes, en la práctica, realizar una transformación de la variable dependiente con el fin de que siga una distribución normal suele ayudar en la distribución de los residuos [8], sin embargo, la suposición de independencia de los residuos comúnmente se viola en los modelos que utilizan datos espaciales. Esto último por la autocorrelación espacial presente en el término de error, lo que significa que el rendimiento del modelo en sí mismo depende de la ubicación geográfica.

Podemos evaluar esto utilizando el índice de Moran, buscamos armar una matriz de pesos, para medir la interacción entre cada tramo censal, para ello generamos una "neighborhood list" en R con el paquete **spdep** usando la función poly2nb(), esta función nos presenta varias formas de catalogar a los vecinos, para el modelo usaremos

- *Contiguity-based neighbors* → se usa cuando la geometría es tipo polygon, las opciones incluyen neighbors tipo "Queen", que se basa en que todos los polygons que compartan un vértice se consideren vecinos, y neighbors tipo "Rook", en donde deben compartir al menos un segmento de linea para ser vecinos, podemos inferir que neighbors tipo "Queen" abre diagonales de manera que tiene menos entradas iguales a cero

```
# Crea neighbor list
nyc_nb <- spdep::poly2nb(nyc_data_prepped, queen = TRUE)
nyc_nb
```

```
Neighbour list object:
Number of regions: 1967
Number of nonzero links: 10700
Percentage nonzero weights: 0.2765509
Average number of links: 5.439756
5 regions with no links:
86 90 242 1727 1767
```

Interpretamos la información de la siguiente manera

- Tenemos 1967 distritos censales en NYC (aquellos con valor NA se omiten)
- El porcentaje de las entradas $w_{i,j}$ tales que $w_{i,j} \neq 0$ es 0.276%
- El número promedio de conexiones entre distritos censales, el promedio de cuantas conexiones posee cada distrito es 5.44
- Existen 5 distritos que no se conectan con nadie.

Podemos hacer un plot de la estructura que posee el objeto “neighborhood list”.

En efecto, observamos la presencia de conexiones diagonales para el modelo “Queen” a diferencia de la estructur de “Rook”, así como en ajedrez, la reina puede moverse en todas direcciones, mientras que la torre solo puede moverse en vertical y horizontal.

Dada la “neighborhood list” creada, se usa para crear una matriz de pesos W usando la función de *spdep* nb2listw(), que nombramos *wts*, con ella se realiza un test de Índice de Moran.

```
# Crea matriz de pesos estandarizada
library(spatialreg)
library(spdep)
set.ZeroPolicyOption(TRUE)
get.ZeroPolicyOption()
wts <- spdep::nb2listw(nyc_nb)
```

Visualizando la matriz de pesos creada, guardado como *wts*

wts[["weights"]]	list [1967]	List of length 1967
[[1]]	double [4]	0.25 0.25 0.25 0.25
[[2]]	double [7]	0.143 0.143 0.143 0.143 0.143 0.143 ...
[[3]]	double [6]	0.167 0.167 0.167 0.167 0.167 0.167
[[4]]	double [2]	0.5 0.5
[[5]]	double [4]	0.25 0.25 0.25 0.25
[[6]]	double [2]	0.5 0.5
[[7]]	double [3]	0.333 0.333 0.333
[[8]]	double [3]	0.333 0.333 0.333
[[9]]	double [5]	0.2 0.2 0.2 0.2 0.2
[[10]]	double [2]	0.5 0.5

Vemos que es una matriz cuyas filas tienen por suma total 1, por ende una matriz de pesos estandarizada, la cual para cada distrito, le da mismo peso a cada vecino que posea.

Con ella podemos realizar un test de índice de Moran, con el fin de verificar si existe autocorrelación espacial en los residuos del modelo, pues dado el caso, la muestra usada en el modelo no cumple ser distribuida identica e independientemente, explicando parte de la deficiencia del modelo de regresión lineal e implicando que el rendimiento del modelo dependa de la ubicación geográfica.

```
# Realiza test de Moran
spdep::moran.test(nyc_data_prepped$residuals, wts)
```

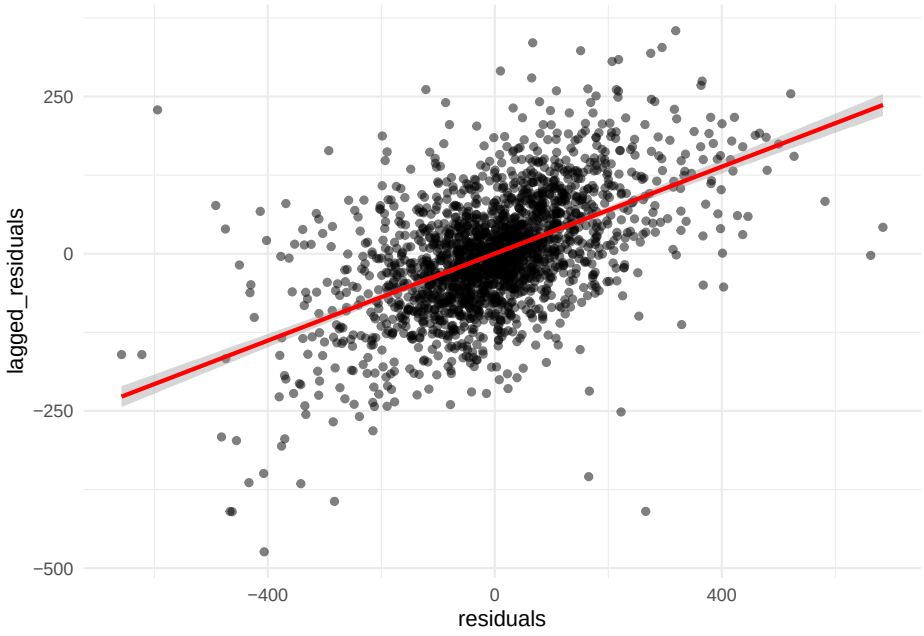
```
        Moran I test under randomisation

data:  nyc_census_data_prepped$residuals
weights: wts  n reduced by no-neighbour observations


Moran I statistic standard deviate = 24.186, p-value < 2.2e-16
alternative hypothesis: greater
sample estimates:
Moran I statistic       Expectation          Variance
     0.3456912899     -0.0005099439      0.0002048902
```

El valor esperado de I bajo la hipótesis nula H_0 de no autocorrelación es de -0.00051, por lo que un valor de $I = 0.345$ es estadísticamente significativo, el p-valor que toma el estadístico es $< 2.2e-16$ lo que nos habla de la significancia que posee en el modelo.

El test de índice de Moran concluye rechazar la hipótesis nula.

Por lo demás, vemos que el índice de Moran tiene un valor positivo, lo que nos dice que en el mapa, distritos censales con atributos similares tienden a agruparse (clusters).

Mediante un *Moran scatterplot*, que consta de un gráfico de datos espaciales comparando los valores de una variable de respuesta Y con sus valores 'lagged', que denotamos $Y_{lag} = WY$, el cual es una versión ponderada de las observaciones vecinas de cada punto, se usa función lag.listw(), junto a la matriz de pesos *wts* y los residuos del modelo.



El plot muestra en el eje horizontal los residuo del modelo, y en el eje vertical, la versión ponderada por la matriz de pesos de los residuos, en donde observamos una **correlacíon positiva**, esto significa que cuando el residuo de un distrito crece, entonces un promedio ponderado de sus alrededores tiende a crecer, en otras palabras, el error del modelo toma valores similares respecto de los valores que toma en sus alrededores, dando cuenta de la autocorrelación en los términos del error, lo que **sugiere rechazar la independencia en los residuos del modelo**.

Motivados por lo anterior, hacemos uso de modelos de regresión que toman en cuenta la autocorrelación espacial presente en los datos, considerando la matriz de pesos que da cuenta de la magnitud de la dependencia espacial.

- Para los códigos utilizados en el modelado de imágenes, ver **Anexo 6**
- Para profundizar en el uso de matrices de peso, ver **Capítulo 7** 'Analyzing US Census Data - Spatial analysis with US Census Data'
- Para complementar en el análisis de datos, ver el **Capítulo 8** 'Analyzing US Census Data - Modeling US Census Data', capítulo donde se estudian conceptos durante la primera sección como indices de segregación y diversidad los cuales son usados en ciencias sociales para explicar patrones demográficos, en la segunda sección se estudian tópicos en modelamiento estadístico, incluyendo métodos de regresión con atributos espaciales, en donde tomamos en cuenta conceptos como la autocorrelación espacial que está inherente en la mayoría de las variables del censo; en la tercera sección del capítulo se estudian conceptos como clasificación, clusterización y regionalización, que son comunes en técnicas de Machine Learning.

Modelos de regresión espacial

Aplicamos cuatro modelos distintos de regresión espacial con los distintos paquetes que proporciona R.

Cada uno de ellos son adaptaciones de distintas formulaciones que consideran una matriz de pesos para hacer frente a los problemas asociados a data espacial.

Consideramos primero dos modelos autorregresivos, los cuales vienen incluidos en el paquete **spatialreg** que se enfocan en añadir términos para abordar la autocorrelación espacial

Spatial Lag Model

El modelo de *lag* espacial tiene en cuenta la dependencia espacial al incluir una variable de retraso sobre el resultado del modelo. Al hacerlo, se tienen en cuenta los efectos de dependencia espacial, es decir, la posibilidad de que los valores en áreas vecinas influyan en los valores en una ubicación determinada. Más precisamente, el modelo tiene la formulación:

$$Y = \alpha + \rho WY + X\beta + \varepsilon$$

Donde formulamos el valor de la respuesta Y como una versión ponderada de los valores que toma la respuesta en los alrededores de cada dato, sumado a una dependencia lineal que tenga la variable Y con los regresores X, tomando W una matriz de pesos estandarizada y ρ refleja la intensidad de la dependencia espacial.

Para visualizarlo más claramente, vemos para cada fila tenemos

$$Y_i = \alpha_i + \rho Y_{lag-i} + \sum_k \beta_k X_{ik} + \varepsilon_i \quad \forall i \in \{1, ..., n\}$$

donde

$$Y_{lag-i} = \sum_j w_{ij} Y_j$$

Por lo que, la variable Y_{lag-i} representa una versión ponderada de los valores que toman las respuestas Y_j que son vecinas de la respuesta Y_i, es decir, aquellos Y_j tales que $w_{ij} \neq 0$.

Podemos ver este modelo como una versión ponderada por por el término $(I - \rho W)^{-1}$ sobre el valor esperado $\alpha + X\beta$ y el residuo del modelo de una regresión lineal clásica [5], basta notar que

$$\begin{aligned} Y &= \alpha + \rho WY + X\beta + \varepsilon \\ \implies Y - \rho WY &= \alpha + X\beta + \varepsilon \\ \implies (I - \rho W)Y &= \alpha + X\beta + \varepsilon \\ \implies Y &= \alpha' + (I - \rho W)^{-1} X\beta + (I - \rho W)^{-1}\varepsilon \end{aligned}$$

Para estimar los parámetros de este modelo, podemos usar el método de máxima verosimilitud, usaremos en este caso funciones incluidas en el paquete *spatialreg*.

Con el paquete **spatialreg** hacemos uso de lagsarlm(), el formato para crear el modelo es similar a los usados en la regresión lineal, usamos como parámetro nuestra matriz de pesos *wts*.

```
lag_model <- spatialreg::lagsarlm(
  formula = formula,
  data = nyc_data_prepped,
  listw = wts,
  zero.policy=TRUE #Importante  omitir valores NA
  )
summary(lag_model, Nagelkerke = TRUE)
```

```
Call:spatialreg::lagsarlm(formula = formula, data = nyc_data_prepped,
listw = wts, zero.policy = TRUE)

Residuals:
      Min        1Q    Median        3Q       Max
-634.1228  -62.9577   -1.1571   60.0997  659.2782

Type: lag
Regions with no neighbours included:
 86 90 242 1728 1768
Coefficients: (asymptotic standard errors)
                  Estimate  Std. Error  z value  Pr(>|z|)
(Intercept)       1.3959e+02  3.5798e+01   3.8995 9.638e-05
median_rooms      5.3798e+01  5.3775e+00  10.0042 < 2.2e-16
median_income     1.3189e-03  1.4271e-04   9.2419 < 2.2e-16
pct_college       8.6781e-01  3.0321e-01   2.8620  0.004209
pct_foreign_born -1.4747e-01  2.6966e-01  -0.5469  0.584463
pct_white         1.0503e+00  1.9994e-01   5.2529 1.497e-07
pct_black         3.4867e-01  1.7197e-01   2.0275  0.042610
pct_hispanic     -8.7095e-03  1.7668e-03  -4.9296 8.242e-07
pct_asian         1.4127e+00  2.5211e-01   5.6036 2.099e-08
median_age       -9.4528e-01  4.9688e-01  -1.9024  0.057114
percent_ooh      -3.1191e+00  2.3721e-01 -13.1492 < 2.2e-16
pop_density       2.0319e-04  2.8196e-04   0.7207  0.471119

Rho: 0.54771, LR test value: 589.69, p-value: < 2.22e-16
Asymptotic standard error: 0.020508
    z-value: 26.707, p-value: < 2.22e-16
Wald statistic: 713.26, p-value: < 2.22e-16

Log likelihood: -12316.63 for lag model
ML residual variance (sigma squared): 14799, (sigma: 121.65)
Nagelkerke pseudo-R-squared: 0.59749
Number of observations: 1969
Number of parameters estimated: 14
AIC: 24661, (AIC for lm: 25249)
LM test for residual autocorrelation
test value: 1.4002, p-value: 0.23669
```

Al igual que el modelo de regresión lineal se nos entregan los valores de α (intercept) y los β_i asociados a cada regresor, junto a sus p-valores, lo que nos permite hacer un análisis de la significancia que tiene cada regresor en el modelo, por ejemplo, para este modelo vemos que la densidad poblacional no es un factor importante en el modelo, al igual que el porcentaje de extranjeros nacidos (pct_foreign_born), mientras que factores como el ingreso promedio (median_income) o el porcentaje de viviendas en el tramo censal que están siendo ocupadas por sus propietarios (percent_ooh) resultan ser cruciales en la predicción, respecto de esto útlimo, vemos que estas relaciones estadísticas se mantienen significativas en comparación al modelo de regresión lineal, mientras que otras como la cantidad de habitaciones promedio en el tramo censal (median_room) aumentan su p-valor.

Además, el modelo calibra el valor de ρ, que refleja la intensidad de la matriz W, este toma un valor positivo y estadísticamente significativo, pues su p-valor es del orden de 2.22×10^{-16}, lo que sugiere la dependencia espacial, el modelo además entrega un test de autocorrelación en los residuos del modelo, en donde indíca un p-valor de 0.227 que habla de la poca significancia que tiene el fenómeno en los residuos del modelo bajo esta nueva formulación.

El argumento 'Nagelkerke = TRUE' calcula un valor de pseudo-R cuadrado, que es mayor al valor correspondiente para el modelo de regresión lineal, con un valor de 0.59.

Los valores de pseudo R-cuadrado no son directamente comparables al R-cuadrado de los modelos de mínimos cuadrados, tampoco se pueden interpretar como la proporción de la variabilidad en la variable dependiente que es explicada por el modelo, más bien, las medidas de pseudo R-cuadrado son medidas relativas entre modelos lineales similares que indican qué tan bien el modelo explica los datos.

Spatial Error Model

En contraste con el modelo de Lag, los modelos de error espacial incluyen un *lag* en el término de error del modelo. Esto está diseñado para capturar procesos espaciales latentes que actualmente no se están teniendo en cuenta en la estimación del modelo y, a su vez, aparecen en los residuos del modelo [8].

El modelo de error espacial se puede escribir de la siguiente manera:

$$Y = \alpha + X\beta + u$$

Donde u es un término autorregresivo de la forma

$$u = \lambda u_{lag} + \varepsilon$$

Y

$$u_{lag} = Wu \implies u = \lambda W u + \varepsilon$$

Es decir, para cada fila

$$Y_i = \alpha + \sum_k \beta_k X_{ik} + \lambda u_{lag-i} + \varepsilon_i$$

Donde

$$u_i = \lambda u_{lag-i} + \varepsilon_i \ , \ u_{lag-i} = \sum_j w_{ij} u_j$$

Por lo que buscamos calibrar λ y u con un modelo de la forma

$$Y_i = \alpha + \sum_k \beta_k X_{ik} + \lambda \sum_j w_{ij} u_j + \varepsilon_i$$

Notemos que llegamos a una formulación similar al caso del modelo Lag, sin embargo buscamos calibrar u que se incluye en el término del error al considerarlo de la forma $u = \lambda W u + \varepsilon$.

Para la calibración de parámetros, hacemos uso de la función errorsarlm de *spatialreg*

```
error_model <- spatialreg::errorsarlm(
  formula = formula,
  data = nyc_data_prepped,
  listw = wts,
  zero.policy=TRUE
)
summary(error_model, Nagelkerke = TRUE)
```

```
Call:spatialreg::errorsarlm(formula = formula, data = nyc_data_prepped,
    listw = wts, zero.policy = TRUE)

Residuals:
      Min        1Q    Median        3Q       Max
-615.6737  -63.4504   -3.1502   59.4877  647.2780

Type: error
Regions with no neighbours included:
 86 90 242 1728 1768
Coefficients: (asymptotic standard errors)
                   Estimate  Std. Error  z value  Pr(>|z|)
(Intercept)       5.9647e+02  4.0882e+01  14.5901 < 2.2e-16
median_rooms      5.9986e+01  6.1124e+00   9.8138 < 2.2e-16
median_income     1.1220e-03  1.6129e-04   6.9563 3.492e-12
pct_college       1.5031e+00  3.7370e-01   4.0223 5.764e-05
pct_foreign_born -4.7362e-01  3.7505e-01  -1.2628  0.206662
pct_white         9.5976e-01  3.1394e-01   3.0572  0.002235
pct_black        -4.9166e-01  3.0898e-01  -1.5912  0.111561
pct_hispanic     -5.9840e-03  1.9980e-03  -2.9950  0.002744
pct_asian         1.2736e+00  3.9161e-01   3.2523  0.001145
median_age       -8.8651e-01  5.6659e-01  -1.5647  0.117665
percent_ooh      -3.1476e+00  2.5688e-01 -12.2535 < 2.2e-16
pop_density      -5.3587e-04  3.5238e-04  -1.5207  0.128333

Lambda: 0.66716, LR test value: 552.3, p-value: < 2.22e-16
Asymptotic standard error: 0.019925
    z-value: 33.484, p-value: < 2.22e-16
Wald statistic: 1121.2, p-value: < 2.22e-16

Log likelihood: -12335.32 for error model
ML residual variance (sigma squared): 14451, (sigma: 120.21)
Nagelkerke pseudo-R-squared: 0.58977
Number of observations: 1969
Number of parameters estimated: 14
AIC: 24699, (AIC for lm: 25249)
```

La calibración entrega un modelo cuyo pseudo-R^2 es de 0.589, el cual es ligeramente menor al pseudo-R^2 del modelo Lag, por lo demás, *spatialreg* entrega los parámetros del modelo en el mismo formato.

Vemos que el valor que toma λ es estadísticamente significativo, con una magnitud de $\lambda = 0.667$ y un bajo p-valor, esto da cuenta nuevamente de la importancia de considerar la autocorrelación espacial del modelo

Los modelos de lag espacial y de error espacial ofrecen enfoques alternativos para tener en cuenta los procesos de autocorrelación espacial al ajustar modelos.

Según Walker [8], respecto de cual de los dos modelos usar, debe considerarse el contexto del tema en estudio, por ejemplo, si los efectos de dependencia espacial están relacionados con las hipótesis que el analista evalúa, como es el caso del efecto de los valores de las viviendas vecinas en torno al valor de una vivienda, se puede preferir un modelo Lag, por otro lado, si hay factores autocorrelacionados espacialmente y que probablemente influyen en la variable de respuesta Y pero son difíciles de medir cuantitativamente, como puede ser la discriminación o el sesgo racial en el mercado de la vivienda, podría ser preferible un modelo de error espacial.

Test de Índice de Moran

Para complementar, podemos aplicar un test de índice de Moran sobre los residuos de ambos modelos de regresión espacial, para ver si resuelven el problema de la dependencia espacial de los errores.

Spatial Lag Model

```
        Moran I test under randomisation

data:  lag_model$residuals
weights: wts  n reduced by no-neighbour observations


Moran I statistic standard deviate = -0.64445, p-value = 0.7404
alternative hypothesis: greater
sample estimates:
Moran I statistic       Expectation          Variance
    -0.0097224085     -0.0005094244      0.0002043731
```

Spatial Error Model

```
        Moran I test under randomisation

data:  error_model$residuals
weights: wts  n reduced by no-neighbour observations


Moran I statistic standard deviate = -3.3594, p-value = 0.9996
alternative hypothesis: greater
sample estimates:
Moran I statistic       Expectation          Variance
    -0.0485344554     -0.0005094244      0.0002043707
```

Ambos modelos reducen el índice de Moran, acercandose a su valor esperando bajo la hipótesis nula H_0 de no autocorrelación, sin embargo, el modelo de error espacial hace un mejor trabajo sobre la autocorrelación espacial en los residuos, esto pues, si bien los modelos toman valores similares de I, el modelo de error espacial asigna un p-valor de 0.99 al estadístico mientras que en el modelo Lag toma un p-valor de 0.7404, eliminando por completo la dependencia espacial en el error.

Comparativa: Regresión Lineal, Lag Model, Error Model

Realizamos una comparativa entre el primer modelo de regresión lineal y los dos modelos de regresión espacial que hemos introducido, usando los paquetes **jtools** y **huxtable** por medio de la función export_summs().

	Model 1	Model 2	Model 3
(Intercept)	504.79 ***	139.59 ***	596.47 ***
	(39.74)	(35.80)	(40.88)
median_rooms	62.09 ***	53.80 ***	59.99 ***
	(6.46)	(5.38)	(6.11)
median_income	0.00 ***	0.00 ***	0.00 ***
	(0.00)	(0.00)	(0.00)
pct_college	1.70 ***	0.87 **	1.50 ***
	(0.36)	(0.30)	(0.37)
pct_foreign_born	-0.63	-0.15	-0.47
	(0.32)	(0.27)	(0.38)
pct_white	2.42 ***	1.05 ***	0.96 **
	(0.24)	(0.20)	(0.31)
pct_black	1.00 ***	0.35 *	-0.49
	(0.21)	(0.17)	(0.31)
pct_hispanic	-0.01 ***	-0.01 ***	-0.01 **
	(0.00)	(0.00)	(0.00)
pct_asian	3.11 ***	1.41 ***	1.27 **
	(0.30)	(0.25)	(0.39)
median_age	-2.17 ***	-0.95	-0.89
	(0.60)	(0.50)	(0.57)
percent_ooh	-4.51 ***	-3.12 ***	-3.15 ***
	(0.28)	(0.24)	(0.26)
pop_density	0.00 **	0.00	-0.00
	(0.00)	(0.00)	(0.00)
rho		0.55 ***	
		(0.02)	
lambda			0.67 ***
			(0.02)
N	1969	1969	1969
R2	0.46	0.63	0.65

*** p < 0.001; ** p < 0.01; * p < 0.05.

Dicha función nos permite exportar un resumen de los tres modelos, indicando los valores de los parámetros ya calibrados, la significancia estadística que estos toman usando el p-valor, además de una comparativa adaptada de R^2 con el fin de medir la calidad de la precisión que tienen los modelos, mostrando una medida de la varianza de la variable dependiente explicada por medio de los regresores.

Respecto de la calidad de predicción según R^2, tenemos en orden de menor a mayor desempeño

- Model 1: Regresión Lineal - lm() $\rightarrow R^2 = 0.46$
- Model 2: Spatial Lag Model - spatialreg::lagsarlm() $\rightarrow R^2 = 0.63$
- Model 3: Spatial Error Model - spatialreg::errorsarlm() $\rightarrow R^2 = 0.65$

Por lo que, ambos modelos de regresión espacial, logran una amplia mejoría respecto del modelo de regresión lineal simple.

Los modelos abordados en las secciones anteriores, tanto la regresión lineal como sus adaptaciones espaciales, estiman relaciones globales entre la variable dependiente y sus regresores $(\beta_i)_{i=1}^m$, es decir, a pesar de que estos modelos dan cuenta de la autocorrelación espacial, asumen que los coeficientes β son iguales para toda ubicación, por lo tanto, no abordan la **heterogeneidad espacial**, ya que los datos que conforman un sector no siguen una distribución idéntica, por lo que es razonable suponer que la relación entre los regresores y la variable dependiente que se observa para toda la región, varíe significativamente entre sectores.

Geographically Weighted Regression (GWR)

El siguiente modelo, aborda la heterogeneidad espacial aprendiendo un conjunto de parámetros β_{ik} asociado a los regresores del modelo en cada ubicación s_i, debido a esto, uno de los inconvenientes de este modelo es el coste computacional, pues para cada regresor, necesitamos entrenar β_{ik} para ubicación s_i.

Mientras que la formulación de un modelo de regresión lineal, el cual tiene la forma

$$Y_i = \alpha_i + \sum_{k=1}^{m} \beta_k X_{ik} + \varepsilon_i$$

Con los β_k idénticos para cada locación s_i, la formulación del modelo de regresión geográficamente ponderado (GWR) para una ubicación s_i dada, se escribe como

$$Y_i = \alpha_i + \sum_{k=1}^{m} \beta_{ik} X_{ik} + \varepsilon_i$$

Donde el intercepto α_i , los regresores $(X_{ik})_{k=1}^{m}$, y los errores ε_i están todos a locación s_i.

Por lo demás, los parámetros β_{ik} serán coeficientes locales para el regresor X_k con ubicación s_i

El coeficiente $\beta(s_h)$ en locación s_h puede entrenarse vía método de mínimos cuadrados ponderados, donde el peso que se le asigna a cada muestra depende de su distancia a s_h, es decir, buscamos encontrar $\beta(s_h)$ tal que

$$\beta(s_h) = \arg\min_{\beta(s_h)} \sum_i w(s_i, s_h)(y(s_i) - x(s_i)^T \beta(s_h))^2$$

donde $y(s_i)$ y $x(s_i)$ son la respuesta y los regresores en locación (s_i) respectivamente, además $w(s_i, s_h)$ es una función que decae con la distancia entre s_i y s_h, por ejemplo, podemos tomar $w(s_i, s_h) = \exp(-\frac{1}{2}||s_i - s_h||_2^2)$ (Jiang, Z 2019) [5]

Para n datos, considerando la notación matricial antes usada, tomando $W = (w(s_i, s_h))_{i,h}^{n}$ como matriz de pesos, y manteniendo implícita la componente espacial, escribimos la calibración de β_h vía mínimos cuadrados ponderados como

$$\beta(s_h) = \arg\min_{\beta(s_h)} \sum_i w_{ih}(Y_i - \sum_{i=1}^{n} X_{ik}\beta_{ih})^2$$

Podemos calibrar los parametros haciendo uso de los paquetes *GWmodel* y *spgwr*, el modelo se basa en el concepto de *kernel bandwidth* (ancho de banda de kernel) para computar un modelo de regresión local en cada ubicación, el cual se basa en un tipo de kernel, fijo o adaptativo, y asignandole una estructura a w_{ij} que decaiga con la distancia.

Un kernel fijo utiliza una distancia de corte para determinar qué observaciones se incluirán en el modelo local para una ubicación dada , mientras que un kernel adaptativo utiliza los vecinos más cercanos a una ubicación dada [8].

Para efectos del código, los tamaños de ancho de banda (un límite de distancia o el número de vecinos más cercanos) pueden ser seleccionados directamente por el usuario; en el paquete *GWmodel*, se nos proporciona la función bw.gwr() que ayuda a elegir un ancho de banda de kernel apropiado mediante validación cruzada.

Para las simulaciones, se hace uso de dicha herramienta, junto a un kernel adaptativo y para la función de pesos, se calcula una función de decaimiento 'bisquare' de la forma

$$w_{ij} = 1 - (\frac{d_{ij}^2}{h^2})^2$$

donde d_{ij} es la distancia entre las observaciones en locación s_i y sus vecinos en locación s_j, como tomamos un kernel de tipo adaptativo, el valor de h varía, y tomará la distancia entre la ubicación s_i y el vecino más lejano a dicha ubicación.

Con el fin de comparar dos calibraciones del mismo modelo, haremos uso de los métodos *spgwr* el cual implementa un enfoque básico de GWR, para un ancho de banda fijo y un esquema de ponderación que viene dado mientras que *GWModel* optimiza dichos parámetros

Modelo GWR: GWModel

Para el método gwr.basic() del paquete GWmodel, debemos consiste convertir el conjunto de datos en un objeto SpatialPolygons sp (no admite objetos sf) y en seleccionar un "bandwidth" (ancho de banda), la estructura del código es la siguiente:

```
# Convierte a Spatial Polygon
nyc_data_prepped_sp <- nyc_data_prepped %>%
  as_Spatial()

# Optimiza bandwidth
bw <- bw.gwr(
  formula = formula,
  data = nyc_data_prepped_sp,
  kernel = "bisquare",
  adaptive = TRUE
  )

  # Entrena el modelo
  gw_model <- gwr.basic(
  formula = formula,
  data = nyc_data_prepped_sp,
  bw = bw,
  kernel = "bisquare",
  adaptive = TRUE
  )

  # Guarda resultados del modelo
  gw_model_results <- gw_model$SDF %>%
  st_as_sf()
```

El modelo nos entrega el siguiente resumen de los resultados

```
*********************************************************************
*          Results of Geographically Weighted Regression            *
*********************************************************************

*********************Model calibration information*********************
Kernel function: bisquare
Adaptive bandwidth: 228 (number of nearest neighbours)
Regression points: the same locations as observations are used.
Distance metric: Euclidean distance metric is used.

****************Summary of GWR coefficient estimates:******************
                       Min.     1st Qu.      Median     3rd Qu.       Max.
Intercept        -1.3915e+02  3.5281e+02  5.2017e+02  6.9042e+02 1107.2631
median_rooms     -5.6062e+01  6.3320e+01  8.8646e+01  1.1650e+02  225.2997
median_income    -1.3865e-03  1.0219e-04  5.7385e-04  1.2785e-03    0.0036
pct_college      -4.5722e+00 -9.8358e-01  5.5569e-01  2.2916e+00    5.4054
pct_foreign_born -7.8207e+00 -2.1959e+00 -1.3727e-01  1.8348e+00    5.0736
pct_white        -4.0832e+00  2.1527e-01  1.2922e+00  2.4962e+00    8.8601
pct_black        -8.7637e+00 -1.4388e+00 -3.5948e-01  6.4875e-01    4.1924
pct_hispanic     -3.1966e-02 -1.0257e-02 -5.3383e-03  6.4731e-04    0.0248
pct_asian        -3.8100e+00  3.5358e-01  1.1029e+00  2.3325e+00   15.3143
median_age       -1.1680e+01 -3.6273e+00 -8.2516e-01  2.0362e+00    7.3398
percent_ooh      -6.7411e+00 -4.2751e+00 -3.1605e+00 -2.3080e+00    1.3729
pop_density      -1.1661e-02 -3.3134e-03 -1.1423e-03  4.5817e-04    0.0088
************************Diagnostic information*************************
Number of data points: 1969
Effective number of parameters (2trace(S) - trace(S'S)): 303.8232
Effective degrees of freedom (n-2trace(S) + trace(S'S)): 1665.177
AICc (GWR book, Fotheringham, et al. 2002, p. 61, eq 2.33): 24436.11
AIC (GWR book, Fotheringham, et al. 2002,GWR p. 96, eq. 4.22): 24140.34
BIC (GWR book, Fotheringham, et al. 2002,GWR p. 61, eq. 2.34): 23694.62
Residual sum of squares: 21641574
R-square value:  0.7213625
Adjusted R-square value:  0.6704926
```

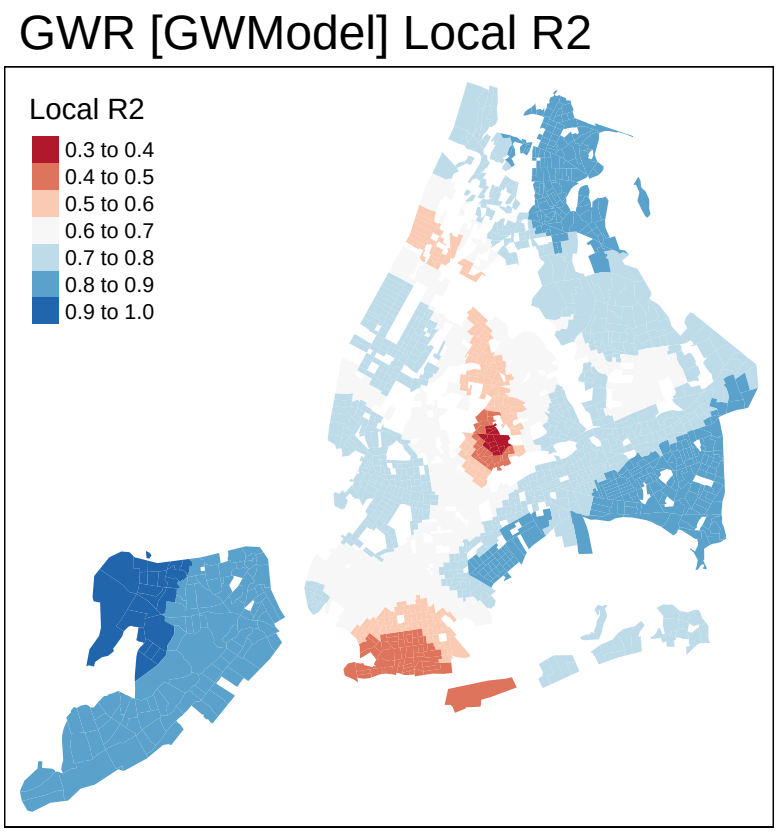
La tabla *Summary of GWR coefficient estimates* entrega las estadísticas entre las que se mueven los parámetros β_{ik}, por otro lado, *Model calibration information* resume el tipo de modelo usado, el cual fue de kernel adaptativo con función de decaimiento *bisquare*, junto a la distancia euclidiana.

Además el modelo entrega una medida de $R^2 = 0.721$, lo que es una mejoría notable en cuanto a la precisión del modelo, comparado con los modelos de lag y error, los cuales no consideran el problema de la **heterogeneidad espacial**

Además, debido a que los coeficientes calibrados tienen asociados una posición geográfica, podemos observar una medida local del R^2 respecto del desempeño del modelo por distritos, y poder visualizar las zonas en donde predice mejor el modelo.

Primero tenemos que añadir los datos de R2 local al objeto sp, después de esto podemos utilizar el paquete *tmap* para crear un mapa para el modelo *GWRmodel*.

```
# Añadimos la data como objeto sp
nyc_data_prepped_sp$gwmodel <- gw_model$SDF$Local_R2

# Print
nyc_gw_model <- tm_shape(nyc_data_prepped_sp) +
  tm_fill("gwmodel",
          palette = "RdBu",
          title = "Local R2") +
  tm_layout(main.title = "GWR [GWModel] Local R2")
nyc_gw_model
```



Modelo GWR: GWModel

Hacemos una comparativa con el modelo calibrado por el paquete *spgwr*, la estructura es la siguiente

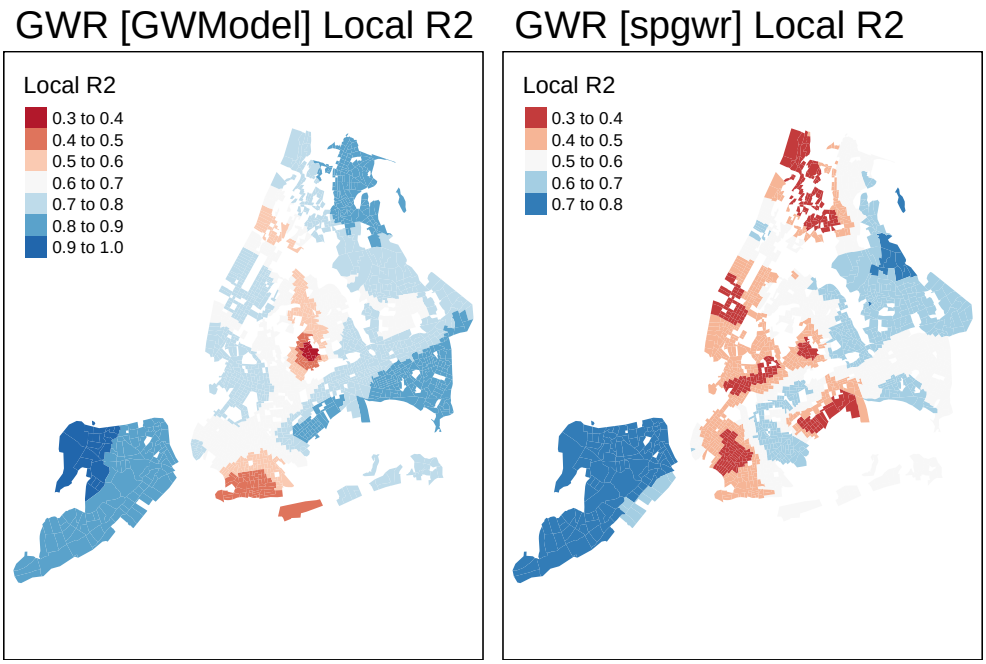
```
# Optimiza bandwitch
bw2 <- gwr.sel(formula = formula,
               data = nyc_data_prepped_sp,
               adapt = TRUE,
               gweight = gwr.bisquare,
               method = "cv",
               verbose = TRUE)

# Entrena modelo
spgwr_model <- gwr(formula = formula,
            data = nyc_data_prepped_sp,
            adapt = bw2,
            gweight = gwr.bisquare,
            hatmatrix = TRUE)
```

Al ejecutar el código, se nos entrega la siguiente tabla resumen del modelo

```
Call:
gwr(formula = formula, data = nyc_data_prepped_sp, gweight = gwr.bisquare,
    adapt = bw2, hatmatrix = TRUE)
Kernel function: gwr.bisquare
Adaptive quantile: 0.1172942 (about 230 of 1969 data points)
Summary of GWR coefficient estimates at data points:
                      Min.     1st Qu.      Median     3rd Qu.        Max.
X.Intercept.     -1.3234e+02  3.5598e+02  5.1797e+02  6.8761e+02  1.1034e+03
median_rooms     -5.4004e+01  6.3359e+01  8.8718e+01  1.1658e+02  2.2349e+02
median_income    -1.3661e-03  1.0472e-04  5.8443e-04  1.2844e-03  3.5768e-03
pct_college      -4.5385e+00 -9.7296e-01  5.5861e-01  2.3061e+00  5.4035e+00
pct_foreign_born -7.7877e+00 -2.2111e+00 -1.5371e-01  1.8219e+00  5.0673e+00
pct_white        -4.0318e+00  2.1770e-01  1.2985e+00  2.4893e+00  8.6795e+00
pct_black        -8.5964e+00 -1.4199e+00 -3.5870e-01  6.2765e-01  4.2056e+00
pct_hispanic     -3.2066e-02 -1.0290e-02 -5.3734e-03  5.6147e-04  2.3899e-02
pct_asian        -3.6405e+00  3.6020e-01  1.0962e+00  2.3491e+00  1.5210e+01
median_age       -1.1543e+01 -3.6504e+00 -7.8090e-01  2.0299e+00  7.3074e+00
percent_ooh      -6.7094e+00 -4.2737e+00 -3.1664e+00 -2.3289e+00  1.3153e+00
pop_density      -1.1493e-02 -3.3111e-03 -1.1427e-03  4.3578e-04  8.7438e-03
                    Global
X.Intercept.     504.7871
median_rooms      62.0936
median_income      0.0021
pct_college        1.6961
pct_foreign_born  -0.6281
pct_white          2.4214
pct_black          1.0034
pct_hispanic      -0.0140
pct_asian          3.1076
median_age        -2.1663
percent_ooh       -4.5130
pop_density        0.0010
Number of data points: 1969
Effective number of parameters (residual: 2traceS - traceS'S): 298.8432
Effective degrees of freedom (residual: 2traceS - traceS'S): 1670.157
Sigma (residual: 2traceS - traceS'S): 114.1081
Effective number of parameters (model: traceS): 227.4935
Effective degrees of freedom (model: traceS): 1741.506
Sigma (model: traceS): 111.7461
Sigma (ML): 105.0926
AICc (GWR p. 61, eq 2.33; p. 96, eq. 4.21): 24435.82
AIC (GWR p. 96, eq. 4.22): 24146.04
Residual sum of squares: 21746532
Quasi-global R2: 0.7200112
```

Al igual que con el método *GWModel*, nos entregan estadísticas respecto de los valores que toman los regresores para las distintas locaciones acompañado de otros coeficientes, como R^2, la suma del cuadrado de los residuos, entre otros parámetros asociados.

El modelo arroja un valor 'cuasi-global' de $R^2 = 0.72$, lo que es imperceptiblemente menor, por lo que procedemos a comparar el desempeño de los modelos respecto de su R^2 local



Notamos que el modelo calibrado por el paquete GWModel predice mejor a nivel general, y existen zonas en donde el método *spgwr* posee un notorio peor desempeño en comparación al método *GWModel*

Geographically Weighted Random Forest

Finalmente, a modo experimental, implementamos el siguiente modelo que aborda la heterogeneidad espacial mediante un conjunto de modelos de bosques aleatorios (RF) localmente calibrados, los modelos RF localmente desarrollados solo usan una cierta cantidad de puntos de datos vecinos para entrenar el modelo [3].

La ecuación para la calibración se formula como

$$Y_i = a(s_i)x(s_i) + \varepsilon$$

Donde s_i referencia la posición geográfica y $x(s_i)$ son los datos de entrenamiento

La diferencia entre un GWR, con una formulación lineal y un GRF, es que podemos modelar la no-estacionariedad de los datos junto con un modelo no lineal, lo que vuelve menos restríctiva la función que queremos modelar.

Para entrenar el modelo hacemos uso del paquete *SpatialML* para aplicar el modelo de bosque aleatorio. Este modelo necesita que se le pase como argumento un objeto de coordenadas, por lo que trabajamos con los centroides de las geometrías.

Este modelo es diferente del clásico random forest, ya que se construye un submodelo para cada ubicación de datos teniendo en cuenta sólo las observaciones cercanas. Estas observaciones cercanas se deciden por la selección del "bandwidth" óptimo. Sin embargo, este proceso consume mucho tiempo, por lo que se escoge aleatoriamente un valor de 98 en bandwidth para no sobrecargar el coste computacional.

Al entrenar el modelo, el paquete entrega el siguiente resumen.

```
Call:
 ranger(formula_grf, data = nyc_grf_prepped, num.trees = 500,      mtry = 2, importance =
"impurity", num.threads = NULL)

Type:                             Regression
Number of trees:                  500
Sample size:                      1969
Number of independent variables:  11
Mtry:                             2
Target node size:                 5
Variable importance mode:         impurity
Splitrule:                        variance
OOB prediction error (MSE):       17707.03
R squared (OOB):                  0.5513357
    median_rooms    median_income      pct_college pct_foreign_born        pct_white
pct_black
       4606936         10389873         12857798          6507840          8863042
7202205
    pct_hispanic        pct_asian       median_age      percent_ooh      pop_density
       3932309          4609792          4978187          5044124          5189713
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-757.5622  -54.9174   -0.1554   -3.0302   54.9276  596.4478
      Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
-146.36310   -4.93200    0.02363   -0.09956    4.74542   98.33541
```

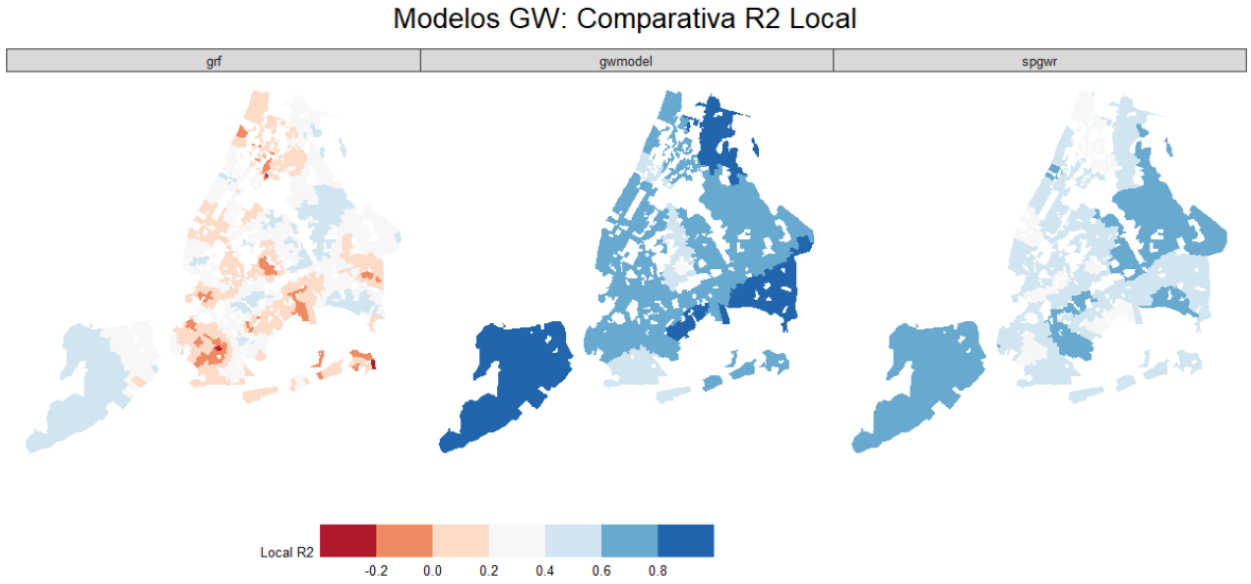
Entregando una medida de $R^2 = 0.551$, que es notoriamente inferior a los modelos clásicos de regresión geográficamente ponderada, sin embargo, sigue entregando mejores resultados que una regresión lineal simple en cuanto a R^2 global.

Para ver el código utilizado en la implementación del modelo, ver **Anexo 7**

Comparativa: R2 Local en modelos GWR y GRF

El modelo GRF trae consigo una medida de R^2 local, por lo que procedemos a comparar gráficamente con respecto del modelo GWR

```
# Añadimos R2 local
nyc_data_prepped_sp$grf <- grf_model$LGofFit$LM_Rsq100 #R2 Local
```



Podemos ver que el modelo *GWR* del paquete *GWmodel* tiene, con diferencia, las puntuaciones R^2 locales más altas, seguidas por el paquete *spgwr* (GWR) y finalmente *SpatialML* (GRF)

También es importante considerar el coste computacional de los modelos, para el caso del modelo GRF, los tiempos de ejecución durante el entrenamiento de los parámetros es notablemente mayor que el del modelo GWR, además, la memoria consumida para almacenar datos en el caso de los métodos *GWModel* y *spgwr* no superan los 40MB, mientras que el modelo *GRF* calibrado por *SpatialML* requiere de 2.3GB

Por lo que, para efectos de los datos usados, el mejor método de predicción para el valor de las viviendas es el modelo GWR, ejecutado por la librería *GWModel*

Para los códigos usados en la visualización de datos, ver **Anexo 7**.

Conclusión

A lo largo del documento, se estudian las problemáticas asociadas al análisis de datos georreferenciados, tales como la autocorrelación espacial y la heterogeneidad espacial inmersa en los datos debido a su naturaleza demográfica.

Se presentan métodos para cuantificar la dependencia espacial, y realizar inferencia estadística en cuanto a este fenómeno, como es el caso del índice de Moran, y se realizan simulaciones con datos de la *American Community Survey (2016-2020)* para ver sus efectos sobre métodos de predicción clásicos, tales como la regresión lineal simple, modelo que no toma en cuenta las componentes espaciales de los datos, las cuales, llevan a rechazar las hipótesis asociadas a este, como es el caso de la independencia entre los residuos del modelo, esto no permite asegurar la calidad de la estimación de los parámetros β, pues dado el Teorema de Gauss-Markov, se necesitan de todas las hipótesis para asegurar que β es el mejor estimador lineal insesgado, es por esta razón que se necesitan modelos que tomen en cuenta la dependencia espacial de los datos, con el fin de obtener una mejor predicción, a partir de esto, se realizan distintos modelos adaptados para la predicción espacial.

Los modelos de Lag y Error, los cuales toman en cuenta la magnitud de la dependencia espacial entre los datos, mediante una matriz de pesos estandarizada, abordan este problema, y logran mejores puntuaciones R^2 que el modelo de regresión lineal simple, donde en particular, el modelo de Error logra eliminar completamente la autocorrelación espacial en los residuos del modelo, sin embargo, sigue presente el problema de la heterogeneidad espacial, motivados por esto, introducimos modelos de regresión geográficamente ponderados (GWR), los cuales ajustan un modelo lineal a cada ubicación de los datos, alcanzando valores R^2 globales más altos que los modelos de Lag y Error, esto pues al calibrar los parámetros para cada ubicación, somos capaces de medir las variaciones locales dentro del modelo, logrando tratar el problema de la heterogeneidad espacial, pues de esta manera, la relación lineal que tienen los regresores con la variable dependiente, no es global, sino que varía dependiendo de la ubicación.

Además, de forma experimental, se aplica un modelo basado en técnicas de Machine Learning, mediante Random Forest (GRF), sin embargo, obtuvo una puntuación menor términos de R^2 respecto de los modelos GWR, y se mantuvo cerca en cuanto a precisión con los modelos de Lag y Error.

Comparativamente, se nota una mejoría con respecto a la precisión de modelos de regresión clásicos. Estos resultados pueden ser útiles para diversos ámbitos de la industria, en cuyas tomas de decisiones intervengan datos que están asociados a una componente geográfica, como es el caso de la predicción de precios de vivienda dentro del mercado inmobiliario.

Referencias

[1] Chen, Y. 2022. Deriving two sets of bounds of moran's index by conditional extremum method. (2022). DOI:https://doi.org/10.48550/ARXIV.2209.08562.

[2] Chen, Y. 2013. New Approaches for Calculating Moran's Index of Spatial Autocorrelation. *PLoS ONE*. 8, 7 (Jul. 2013), e68336. DOI:https://doi.org/10.1371/journal.pone.0068336.

[3] Georganos, S. and Kalogirou, S. 2022. A Forest of Forests: A Spatially Weighted and Computationally Efficient Formulation of Geographical Random Forests. *ISPRS International Journal of Geo-Information*. 11, 9 (Aug. 2022), 471. DOI:https://doi.org/10.3390/ijgi11090471.

[4] Ikramov, K.D. 1994. A simple proof of the generalized Schur inequality. *Linear Algebra and its Applications*. 199, (Mar. 1994), 143–149. DOI:https://doi.org/10.1016/0024-3795(94)90346-8.

[5] Jiang, Z. 2019. A survey on spatial prediction methods. *IEEE Transactions on Knowledge and Data Engineering*. 31, 9 (2019), 1645–1664. DOI:https://doi.org/10.1109/TKDE.2018.2866809.

[6] Nikparvar, B. and Thill, J.-C. 2021. Machine Learning of Spatial Data. *ISPRS International Journal of Geo-Information*. 10, 9 (Sep. 2021), 600. DOI:https://doi.org/10.3390/ijgi10090600.

[7] Tobler, W.R. 1970. A computer movie simulating urban growth in the detroit region. *Economic Geography*. 46, (Jun. 1970), 234. DOI:https://doi.org/10.2307/143141.

[8] Walker, K. 2023. Analyzing US census data. (Jan. 2023). DOI:https://doi.org/10.1201/9780203711415.

Anexo

Anexo 1: Librerías

Las librerías durante la realización del proyecto son las siguientes:

```
library(tidycensus)
library(tidyr)
library(tidyverse)
library(censusapi)
library(tmap)
library(ggplot2)
library(dplyr)
library(stringr)
library(units)
library(stats)
library(grDevices)
library(dotenv)
library(sf)
library(corrr)
library(spatialreg)
library(spdep)
library(jtools)
library(huxtable)
library(GWmodel)
library(spgwr)
library(SpatialML)
library(ggthemes)
```

Para más información respecto de las librerías, consultar Analyzing US Census Data - Kyle Walker

Anexo 2: Activacion API key

Para hacer uso de la data que nos proporciona el paquete tidycensus debemos hacer uso de una *key* que nos permita el acceso a la información.

El siguiente código ejecuta la activación de la llave 'API Key' que nos permite descargar Census Data, mediante funciones como get_acs, el primer argumento es la llave utilizada en este código.

```
census_api_key("6034739b488f5fc230e467601ed20256bb25831b", install = TRUE)
```

Este comando tiene la estructura

```
census_api_key(key, overwrite = BOOL, install = BOOL)}
```

Argumentos

- key: La API Key entregada por el Censo, ingresar con " ". Se obtiene en API Census.
- overwrite: Si está en TRUE, sobreescribirá sobre una ya existente CENSUS_API_KEY que tengamos instalado en nuestro archivo .Renviron
- install: Si está en TRUE, instalará la llave en nuestro archivo .Renviron para las futuras sesiones, de no existir crea uno. Viene en FALSE por defecto.

Despues de instalada la llave, puede usarse en cualquier momento llamando el siguiente comando

```
Sys.getenv("CENSUS_API_KEY")
```

```
## [1] "6034739b488f5fc230e467601ed20256bb25831b"
```

Reload del enviroment para poder usar la llave sin tener que resetear R

```
readRenviron("~/.Renviron")
```

Anexo 3: Uso de tidycensus para la obtención y manejo de datos

Para obtener datos de las distintas bases *tidycensus* ofrece las siguientes funciones

- get_decennial() : Solicita datos de las API US Decennial Census para 2000, 2010 y 2020.
- get_acs() : Solicita datos de las muestras de la American Community Survey de 1 y 5 años. Los datos están disponibles desde el ACS de 1 año hasta 2005 y el ACS de 5 años hasta 2005-2009.
- get_estimates() : Interfaz para las Population Estimates APIs. Estos conjuntos de datos incluyen estimaciones anuales de las características de la población por estado, condado y área metropolitana, junto con componentes de estimaciones demográficas de cambio como nacimientos, muertes y tasas de migración.
- get_pums() : Accede a los datos de ACS Public Use Microdata Sample APIs, Estas muestras incluyen registros anónimos a nivel individual de la ACS organizados por hogar y son muy útiles para muchos análisis de ciencias sociales, get_pums() se cubre con más profundidad en los Capítulos 9 y 10 de *Analyzing US Census data*.
- get_flows() : Interfaz para la ACS Migration Flows APIs. Incluye información sobre los flujos de entrada y salida de varias geografías para las muestras de ACS de 5 años, lo que permite realizar análisis de origen y destino

El código utilizado para elegir las variables necesarias para los modelos de predicción es el siguiente:

```
#package
library(dotenv)
library(sf)
library(tidycensus)
library(dplyr)

# Variable: Condados que forman NYC
nyc_counties <- c("Bronx","Kings","New York","Queens","Richmond")

# Lista de regresores a utilizar
variables_to_get <- c(
  median_value = "B25077_001",
  median_rooms = "B25018_001",
  median_income = "DP03_0062",
  total_population = "B01003_001",
  median_age = "B01002_001",
  pct_college = "DP02_0068P",
  pct_foreign_born = "DP02_0094P",
  pct_white = "DP05_0077P",
  pct_black = "DP05_0078P",
  pct_hispanic = "DP05_0070P",
  pct_asian = "DP05_0080P",
  percent_ooh = "DP04_0046P"
)
```

Para ver que variables se pueden obtener, *tidycensus* nos provee de la función load_variables(), dicha función requiere de 2 argumentos, *year* que toma el año de referencia de la data, y *dataset*.

Para el Decennial Census 2000 a 2010, usar "sf1" o "sf2", el 2020 Decennial Census tambien acepta "sf3" y "sf4", sf hace referencia a Summary Files.

Para variables de la American Community Survey, debemos especificar el año de la encuesta, por ejemplo "acs1" para el primer año de la ACS, por ejemplo si se quiere acceder a la data *5-year ACS*

```
load_acs = load_variables(year = 2020, dataset = "acs5")
```

	name	label	concept	geography
1	B25034_001	Estimate!!Total:	YEAR STRUCTURE BUILT	block group
2	B25034_002	Estimate!!Total:!!Built 2014 or later	YEAR STRUCTURE BUILT	block group
3	B25034_003	Estimate!!Total:!!Built 2010 to 2013	YEAR STRUCTURE BUILT	block group
4	B25034_004	Estimate!!Total:!!Built 2000 to 2009	YEAR STRUCTURE BUILT	block group
5	B25034_005	Estimate!!Total:!!Built 1990 to 1999	YEAR STRUCTURE BUILT	block group
6	B25034_006	Estimate!!Total:!!Built 1980 to 1989	YEAR STRUCTURE BUILT	block group
7	B25034_007	Estimate!!Total:!!Built 1970 to 1979	YEAR STRUCTURE BUILT	block group
8	B25034_008	Estimate!!Total:!!Built 1960 to 1969	YEAR STRUCTURE BUILT	block group
9	B25034_009	Estimate!!Total:!!Built 1950 to 1959	YEAR STRUCTURE BUILT	block group
10	B25034_010	Estimate!!Total:!!Built 1940 to 1949	YEAR STRUCTURE BUILT	block group
11	B25034_011	Estimate!!Total:!!Built 1939 or earlier	YEAR STRUCTURE BUILT	block group
12	B08604_001	Estimate!!Total:	WORKER POPULATION FOR WORKPLACE GEOGRAPHY	county
13	C18121_001	Estimate!!Total:	WORK EXPERIENCE BY DISABILITY STATUS	tract
14	C18121_002	Estimate!!Total:!!Worked full-time, year round:	WORK EXPERIENCE BY DISABILITY STATUS	tract
15	C18121_003	Estimate!!Total:!!Worked full-time, year round:!!With a disabi...	WORK EXPERIENCE BY DISABILITY STATUS	tract
16	C18121_004	Estimate!!Total:!!Worked full-time, year round:!!No disability	WORK EXPERIENCE BY DISABILITY STATUS	tract
17	C18121_005	Estimate!!Total:!!Worked less than full-time, year round:	WORK EXPERIENCE BY DISABILITY STATUS	tract
18	C18121_006	Estimate!!Total:!!Worked less than full-time, year round:!!Wit...	WORK EXPERIENCE BY DISABILITY STATUS	tract
19	C18121_007	Estimate!!Total:!!Worked less than full-time, year round:!!No ...	WORK EXPERIENCE BY DISABILITY STATUS	tract
20	C18121_008	Estimate!!Total:!!Did not work:	WORK EXPERIENCE BY DISABILITY STATUS	tract
21	C18121_009	Estimate!!Total:!!Did not work:!!With a disability	WORK EXPERIENCE BY DISABILITY STATUS	tract
22	C18121_010	Estimate!!Total:!!Did not work:!!No disability	WORK EXPERIENCE BY DISABILITY STATUS	tract

Más detalles en el capítulo 2 del libro Analyzing US Census Data

Anexo 4: Uso de librerías ggplot2 y tmap para plots con referencias geográficas

Dada la naturaleza del problema de predicción georeferenciada, nos interesa visualizar la data junto a su componente espacial, a continuación se presenta un breve manual de uso de las librerías *ggplot2* y *tmap* para visualizar los datos acompañados de sus referencias geográficas, los cuales en R se codifican como objetos tipo *polygon*, usando un ejemplo con datos provenientes de NYC.

NY US Census data

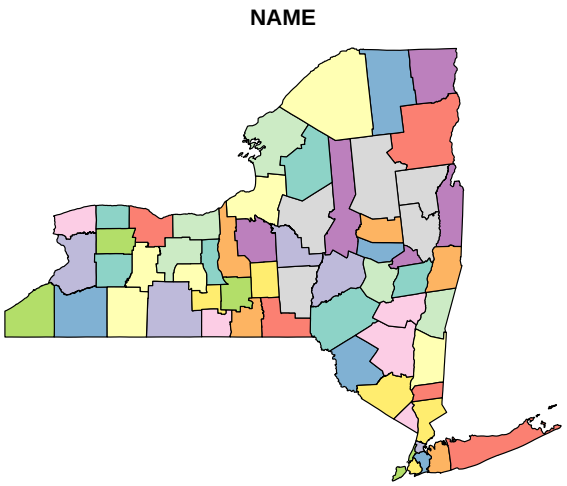
Tomamos como caso de prueba al estado de New York, para visualizar el valor medio de las viviendas a nivel de condados, podemos variar el nivel geográfico con el parámetro *geography* (Walker K., 2023) [8]

```
# Creamos el objeto median_nyc que contiene la variable "median income"
# con nivel geografico condados(county).
medianincomenystate <- get_acs(geography = "county",
        state = "New York",
        geometry = TRUE, #descarga el componente espacial del tramo censal
        variable = "B19013_001" #median income
        )
```

Plot del estado New York

Comencemos a visualizar la información, probaremos primeramente con plot, y luego usaremos herramientas mas avanzadas que nos ofrecen los paquetes

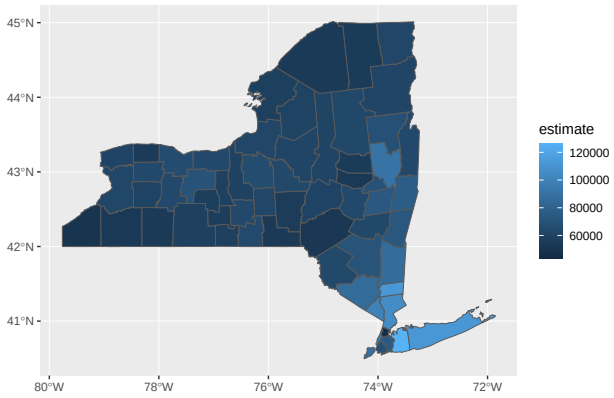
```
plot(medianincomenystate["NAME"])
```



Plot del valor medio de las viviendas en New York, map-making con ggplot2 y geom_sf

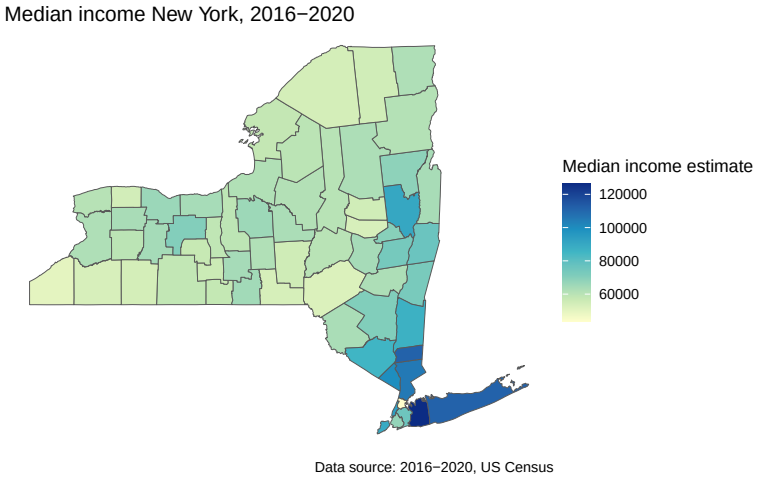
En *ggplot2* podemos plotear rapidamente objetos de tipo *sf* mediante geom_sf(), para entender la sintaxis realizamos el siguiente plot de el estimado de la variable "Median income New York".

```
ggplot(data = medianincomenystate, aes(fill = estimate)) +
  geom_sf()
```



Podemos customizar nuestros plots en ggplot2, la estructura es la siguiente

```
ggplot(data = medianincomenystate, aes(fill = estimate)) +
  geom_sf() +
  scale_fill_distiller(palette = "YlGnBu",
                       direction = 1) +
  labs(title = "Median income New York, 2016-2020",
       caption = "Data source: 2016-2020, US Census",
       fill = "Median income estimate") +
  theme_void()
```

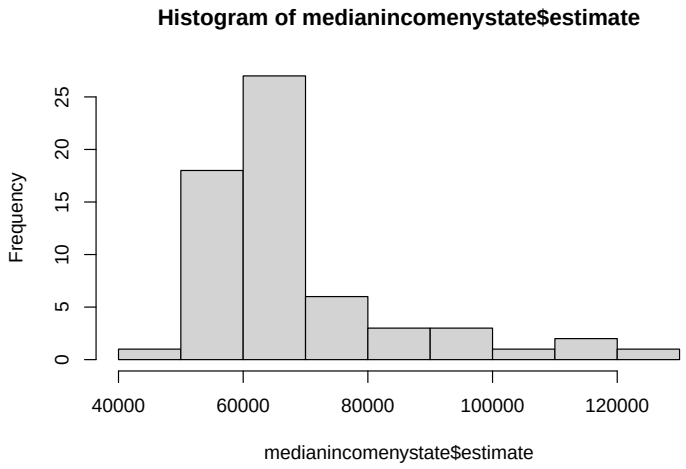


Las funciones que acompañan la sintaxis nos permiten customizar nuestro plot en ggplot2.

- scale_fill_distiller() : Nos permite especificar una paleta de colores de ColorBrewer en el plot.
- labs() : Nos permite añadir título, caption, legend label en el plot.
- theme_void() : Nos permite remover el fondo y la grilla cuadrícular.

Histograma del valor medio de las viviendas en el estado New York

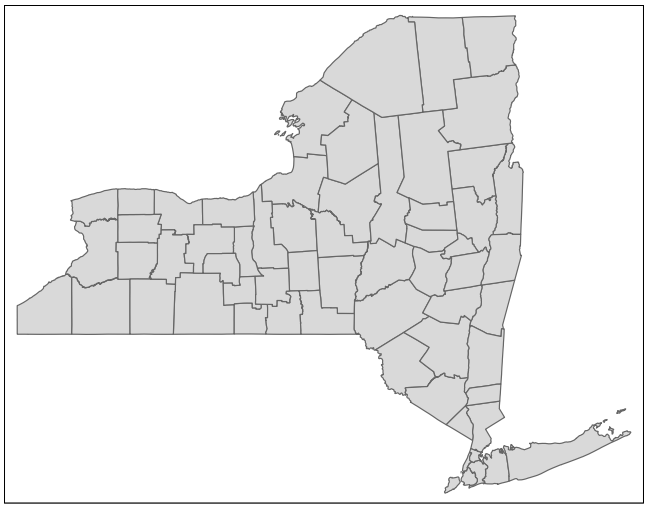
```
hist(medianincomenystate$estimate)
```



Map-making con tmap

La sintaxis en similar a la usada en *ggplot2*, el objeto mapa se inicializa con la función tm_shape() y nos permite visualizar los distritos censales con tm_polygons()

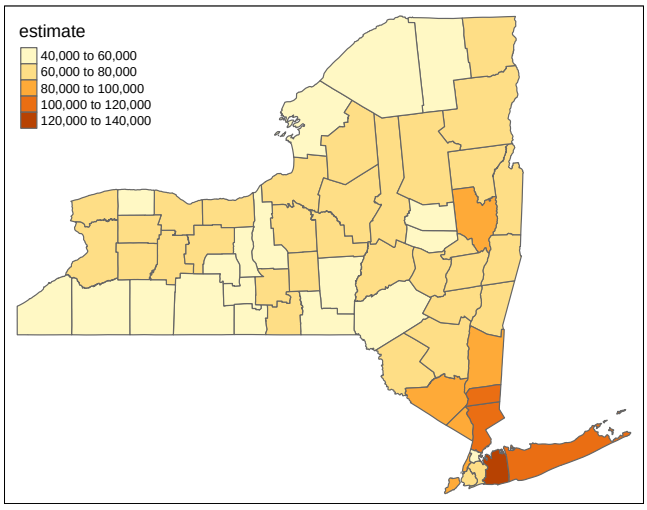
```
library(tmap)
tm_shape(medianincomenystate) +
  tm_polygons()
```



Variables en tmap

Veamos nuevamente la variable "median income", para ello llamamos la variable a visualizar dentro de tm_polygons

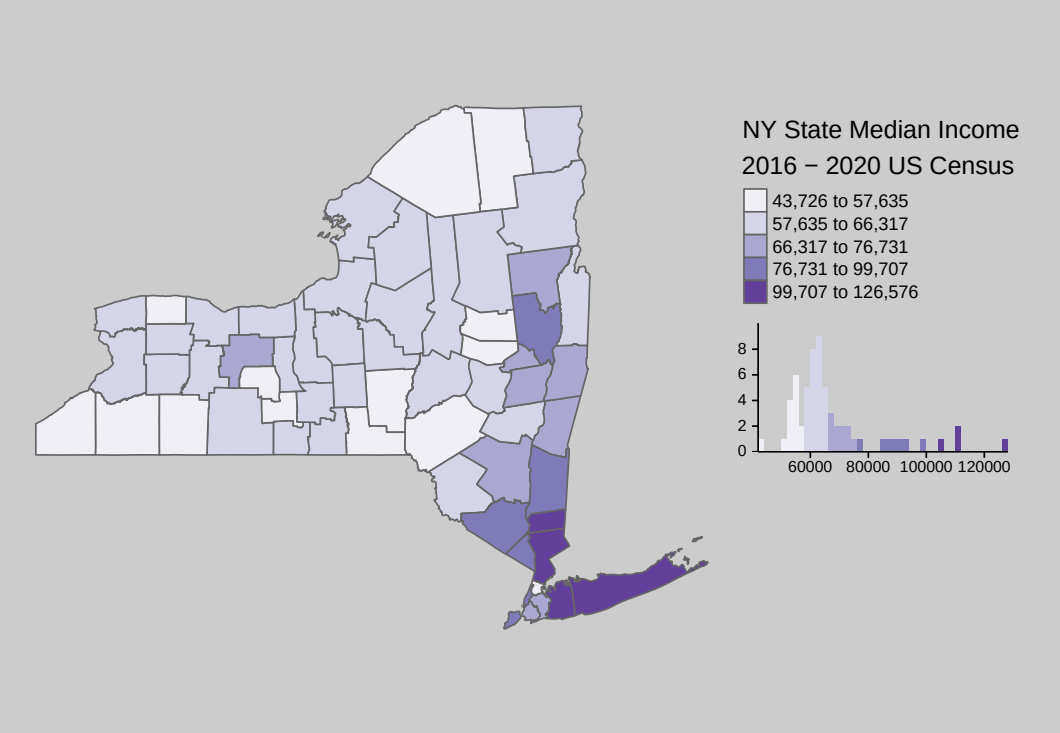
```
library(tmap)
tm_shape(medianincomenystate) +
  tm_polygons(col = "estimate")
```



Labels y otras opciones de diseño

Podemos añadir más variables y seguir personalizando nuestros plots, por ejemplo añadiendo histogramas por distintos tipos de clasificación, quantiles ("quantile"), equal intervals ("equal") y Jenks natural breaks ("jenks"), con tm_layout() nos permite customizar el estilo del mapa, del histograma y añadir leyendas.

```
tm_shape(medianincomenystate) +
  tm_polygons(col = "estimate",
              style = "jenks",
              n = 5, #num de intervalos
              palette = "Purples",
              title = "2016 - 2020 US Census",
              legend.hist = TRUE) +
  tm_layout(title = "NY State Median Income",
            frame = FALSE,
            legend.outside = TRUE,
            bg.color = "grey80", #backgroundcolor
            legend.hist.width = 7)
```



NYC Median Income

Podemos trabajar con niveles geográficos de menor nivel, como condados y tracts, trabajamos la ciudad de New York, formado por ciertos condados.

```
# Creamos el objeto medianincomenyc que contiene la variable "median income"
# con nivel geografico de tract.
medianincomenyc <- get_acs(geography = "tract",
        state = "New York",
        geometry = TRUE, #descarga el componente espacial del tramo censal
        variable = "B19013_001" #median income
        )
```

Filtramos respecto de los condados que queremos visualizar, aquellos cerca de la ciudad de NY, usamos *tidyr* con la función separate(), de manera de poder filtrar los condados que nos interesan

El siguiente código nos permite crear nuevas columnas "tract" y "county", de manera que podemos filtra aquellos condados de interes con otras funciones, ademas usamos na.omit() para botar aquellos valores con NA y así limpiar la data

```
medianincomenyc <- separate(medianincomenyc,
         NAME,
         into = c("tract", "county"),
         sep = ", ")

medianincomenyc <- medianincomenyc %>% filter(grepl('Bronx County|New York County|Queens County

medianincomenyc <- na.omit(medianincomenyc)
```

Plot con tmap de NYC sobre los ingresos promedios

```
tm_shape(medianincomenyc) +
  tm_polygons(col = "estimate",
              style = "equal",
              palette = "Purples",
              title = "2016 - 2020 US Census",
              legend.hist = TRUE) +
  tm_layout(title = "NYC Median Income by Census Tract",
            frame = FALSE,
            legend.outside = TRUE,
            bg.color = "grey80", #backgroundcolor
            legend.hist.width = 7)
```

Más detalles en el capítulo 6 de Analyzing US Census Data. Ver también Tmap Book

Anexo 5: Discusión sobre el índice de Moran

Veamos que, bajo ciertas condiciones de la matriz W, se tiene que $I \in [-1, 1]$, en (Chen, Y. 2022)[1] tenemos una demostración de que el índice de Moran toma dichos valores cuando W clasifica como "globally normalized wight matrix", pero no entra en detalles, además, se estudia que en la práctica muy típicamente para estas matrices, $I \in (-1, 1)$, en el manual de PQStat se habla de "Spatial weights matrix" como una matriz con coeficientes positivos y filas estandarizadas de manera las filas sumen uno, esto es $\sum_{j=1}^{n} w_{ij} = 1 \ \ \forall i \in \{1, \ldots, n\}$ (*row-normalized*).

En la discusión "Why is Moran's I coming out greater than 1" - StackExchange se muestra un contraejemplo de matriz W cuya suma de las entradas es $\sum_{i,j}^{n} w_{ij} = 1$ y para muestras $(X_1, \ldots, X_4)$, donde sucede que $I = 3$, por lo que si W no cumple la propiedad de ser $row - normalize$, no se puede saber a primeras los valores que puede tomar I, y por tanto, no tenemos una referencia clara de la significancia que pueda tener la magnitud de I en el modelo si no conocemos sus valores máximos y mínimos.

Veremos que se puede relajar la condición de que la suma de las filas sean todas iguales a 1 y pediremos que la suma de las filas sea un número α para cada fila, trabajaremos entonces con matrices de pesos estandarizadas que son las adecuadas para un modelo de autocorrelación

Presentamos una demostración en la que no usaremos técnicas de cálculo ni de optimización para probar que para W una matriz de pesos estandarizada, tenemos que $I \in [-1, 1]$

Haremos uso de los siguientes resultados conocidos

- **Teorema:** Sea A una matriz cuadrada simétrica a coeficientes reales, tenemos que A es ortogonalmente diagonalizable, es decir, podemos escribir $A = PDP^T$, con P matriz ortogonal cuyos vectores columna son los vectores propios de A y D es matriz diagonal con $(d_{ii})_{i=1}^{n} = (\lambda_i)_{i=1}^{n}$ valores propios de A
- **Lema:** Si P es ortogonal, entonces $P^T = P^{-1}$ y además, P es una isometría en $(\mathbb{R}^n, ||\cdot||_2)$, es decir, $||Pw||_2 = ||w||_2 \quad \forall w \in \mathbb{R}^n$

Con ello, podemos probar el siguiente teorema

- **Teorema:** Se tiene que el operador $\lambda_{max} : S^n \to \mathbb{R}$, que asocia a cada $A \in S^n$ matriz simétrica su mayor valor propio $\lambda_{max}(A)$, es convexa, más aún, $\lambda_{max}(A) = \sup_{||v||=1} v^T A v$ supremo de funcionales lineales sobre S^n.

El resultado anterior es conocido es corolario de los cocientes de Rayleigh, para una demostración detallada de ver Teorema 55 en The Rayleigh's principle and the minimax principle for the eigenvalues of a self-adjoint matrix

Además haremos uso del siguiente lema

- **Lema:** El mayor valor propio de una matriz A con coeficientes no negativos está acotada por la mayor suma de las filas, esto es, $\lambda_{max}(A) \leq \max_i \sum_{j=1}^{n} a_{ij}$.
- **Demostración del lema**: Sea λ un valor propio de una matriz A no negativa asociada a un vector x, tenemos entonces que

$$Ax = \lambda x \implies \lambda|x_i| = |\sum_{j=1}^{n} a_{ij}x_j| \quad \forall i \in \{1, \ldots, n\}$$

$$\implies \lambda|x_i| = |\sum_{j=1}^{n} a_{ij}x_j| \leq (\sum_{j=1}^{n} |a_{ij}|) \max_j |x_j| \quad \forall i \in \{1, \ldots, n\}$$

Usando que $a_{ij} \geq 0 \quad \forall i, j \in \{1, \ldots, n\}$ y que la desigualdad se cumple para todo i tenemos

$$\lambda \max_i |x_i| \leq \max_i (\sum_{j=1}^{n} a_{ij} \max_j |x_j|) = \max_i (\sum_{j=1}^{n} a_{ij}) \max_j |x_j|$$

$$\implies \lambda \leq \max_i (\sum_{j=1}^{n} a_{ij})$$

Como λ era arbitrario deducimos que

$$\lambda_{max}(A) \leq \max_i (\sum_{j=1}^{n} a_{ij})$$

Enunciamos entonces

- **Propiedad:** Para $I = \frac{n}{W}\frac{\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(y_i-\bar{y})(y_j-\bar{y})}{\sum_{i=1}^{n}(y_i-\bar{y})^2}$ donde $\bar{y} = \sum_{i=1}^{n} y_i/n$ con W matriz de pesos estandarizada e $y \in \mathbb{R}^n$, se cumple que $I \in [-1, 1]$.
- **Demostración:** Consideremos el operador $\Phi : \mathbb{R}^n \to \partial B(0,1)$ definido por $\Phi(y) = z = (z_i)_{i=1}^n = (\frac{(y_i-\bar{y})}{\sqrt{(\sum_{i=1}^{n}(y_i-\bar{y})^2)}})_{i=1}^n$

Está bien definida pues $\forall y \in \mathbb{R}^n$

$$||\Phi(y)||_2 = ||z||_2 = \sum_{i=1}^{n} |z_i|^2 = \sum_{i=1}^{n} \frac{(y_i-\bar{y})^2}{\sum_{i=1}^{n}(y_i-\bar{y})^2}$$

$$= \frac{\sum_{i=1}^{n}(y_i-\bar{y})^2}{\sum_{i=1}^{n}(y_i-\bar{y})^2} = 1$$

Notamos que

$$I = \frac{n\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(y_i-\bar{y})(y_j-\bar{y}))}{W(\sum_{i=1}^{n}(y_i-\bar{y})^2)}$$

$$= \frac{n}{W}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\frac{(y_i-\bar{y})(y_j-\bar{y})}{\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2}\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2}}$$

$$\frac{n}{W}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\Phi(y)_i\Phi(y)_j = \frac{n}{W}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}z_iz_j$$

Definimos $V = (\frac{w_{ij}}{W})_{i,j}^n$ matriz normalizada tal que $\sum_{i,j} v_{ij} = 1$, además, como W es matriz de pesos estandarizada, tenemos que V también lo es, por lo que es simétrica, no negativa, y row-normalized.

Entonces, recordando que $||z||_2 = 1$

$$I = n(z^TVz) \leq n \sup_{||z||=1} z^TVz = n\lambda_{max}(V)$$

$$\implies |I| \leq n|\lambda_{max}(V)| \leq n|\max_i(\sum_{j=1}^{n} v_{ij})| \leq n\max_i(\sum_{j=1}^{n}|v_{ij}|)$$

Por un lado, tenemos que $|v_{ij}| = v_{ij}$ y como $\sum_{i=1}^{n}\sum_{j=1}^{n} v_{ij} = 1$ y V es *row-normalized* tenemos que se cumple $\sum_{j=1}^{n} v_{ij} = \frac{1}{n} \quad \forall i \in \{1, \ldots, n\}$, y por lo tanto

$$\implies |I| \leq n\max_i(\sum_{j=1}^{n} v_{ij}) = n * \frac{1}{n} = 1$$

$$\implies I \in [-1, 1]$$

En caso de que W no cumpla la propiedad *row-normalized* de igual manera podemos encontrar una cota para I y esta es $I \in [-n, n]$ donde n es la cantidad total de datos haciendo uso de la desigualdad de Schur [4] que nos dice

- **Teorema:** Sea A una matriz $n \times n$ con valores propios $(\lambda_1, \ldots, \lambda_n)$, entonces se cumple

$$\sum_{i=1}^{n} |\lambda_i|^p \leq \sum_{i,j=1}^{n} |a_{ij}|^p \quad , 1 \leq p < 2$$

Luego, como vimos antes, tenemos ahora V una matriz normalizada, pero no necesariamente *row-normalized*, luego

$$I = n(z^T V z)$$

Y como V es simétrica, la escribimos como $V = P^T D P$ usando el teorema espectral, y como P es ortogonal, entonces $x = Pz$ tiene norma $||Pz||_2 = ||x||_2 = 1$, en particular $|x_i|^2 \leq 1 \ \ \forall i \in \{1, \ldots, n\}$ luego

$$I = n(z^T P^T D P z) = n(x^T D x) = n \sum_{i=1}^{n} \lambda_i(V) |x_i|^2$$

$$\implies |I| \leq n \sum_{i=1}^{n} |\lambda_i(V)| \leq n \sum_{i,j=1}^{n} |v_{ij}| = n$$

$$\implies I \in [-n, n]$$

Retomando el ejemplo visto en "Why is Moran's I coming out greater than 1" - StackExchange donde $I = 3$, vemos que $n = 4$ y entonces $I \in [-n, n]$.

Apesar de este resultado, usamos matrices de peso estandarizadas pues son las que mejor se ajustan al modelo, y donde sabemos que el índice de Moran se mueve entre $[-1, 1]$.

Si sabemos el intervalo en el que se mueve I para nuestro modelo, podemos saber que tan significativo es el valor, y por lo tanto que tan presente esta el fenómeno de autocorrelación espacial.

Respecto del valor esperado del índice de Moran bajo cuando suponemos que no hay autocorrelación espacial, consideramos la siguiente hipótesis nula:

$$H_0 : \text{No existe autocorrelación espacial}$$

Dicha hipótesis consiste en suponer que, dada una permutación de los datos, es decir, cambiando el orden en el que vienen los datos $(x_i)_{i=1}^{N}$, y fijando la matriz de pesos antes del reordenamiento, no se debiese tener un impacto sobre el valor de I.

Por lo tanto, para un muestreo $x = (x_i)_{i=1}^N$, tomamos una permutación $\pi \in S^N$ que aplica sobre el conjunto $(x_i)_{i=1}^N$ con S^N el grupo simétrico.

Consideramos nuestro espacio de probabilidad $(\Omega, \mathcal{F}, \mathbb{P}) = (S^N, \mathcal{P}(S^N), \mathbb{P}_u)$ con $\mathbb{P}_u$ distribución uniforme.

Tenemos entonces

$$\mathbb{E}_\pi(I) = \frac{N}{W} \frac{\sum_{i=1}^N \sum_{j=1}^N w_{ij}((\pi x)_i - \pi\bar{x})((\pi x)_j - \pi\bar{x})}{\sum_{i=1}^N ((\pi x)_i - \pi\bar{x})^2}$$

Claramente, $\pi\bar{x} = \bar{x}$, además haciendo la suma sobre i tenemos

$$\sum_{i=1}^N ((\pi x)_i - \pi\bar{x})^2 = \sum_{i=1}^N (x_i - \bar{x})^2$$

Vemos entonces que

$$\mathbb{E}_\pi[((\pi x)_i - \pi\bar{x})((\pi x)_j - \pi\bar{x})] = \mathbb{E}_\pi[((\pi x)_i - \bar{x})((\pi x)_j - \bar{x})]$$

Sea $f : S^N \to \mathbb{R}$ definida por $f(\pi) = ((\pi x)_i - \bar{x})((\pi x)_j - \bar{x})$, luego tenemos

$$\mathbb{E}_\pi(f(\pi)) = \sum_{\pi_0 \in S^n} f(\pi_0)\mathbb{P}(\pi = \pi_0)$$

$$= \sum_{\pi_0 \in S^N} f(\pi)\frac{1}{N!} = \frac{1}{N!}\sum_{\pi \in S^N} ((\pi x)_i - \bar{x})((\pi x)_j - \bar{x})$$

Como π es una permutación, entonces $(\pi x)_i \neq (\pi x)_j$ para $i \neq j$ y además notamos que para (x_i, x_j) fijos existen $(N-2)!$ permutaciones π tales que $(x_i, x_j) = ((\pi x)_i, (\pi x)_j)$ para $i \neq j$, por lo tanto

$$\sum_{\pi \in S^N} ((\pi x)_i - \bar{x})((\pi x)_j - \bar{x}) = (N-2)! \sum_{i \neq j} (x_i - \bar{x})(x_j - \bar{x})$$

Y entonces

$$\mathbb{E}_\pi[((\pi x)_i - \pi\bar{x})((\pi x)_j - \pi\bar{x})] = \frac{(N-2)!}{N!} \sum_{i \neq j} (x_i - \bar{x})(x_j - \bar{x})$$

$$= \frac{1}{N(N-1)}[(\sum_k x_k - \bar{x})^2 - \sum_k (x_k - \bar{x})^2]$$

$$= \frac{-\sum_k (x_k - \bar{x})^2}{N(N-1)}$$

Luego

$$\mathbb{E}_\pi(I) = \frac{N}{W} \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} (\frac{-\sum_k (x_k - \bar{x})^2}{N(N-1)})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

Y por lo tanto

$$\mathbb{E}_\pi(I) = -\frac{1}{N-1}$$

Concluimos que de aceptar la hipótesis nula, se espera que

$$\mathbb{E}(I) = -\frac{1}{N-1}$$

Anexo 6: Códigos para plots de modelos para generación de matriz de pesos estandarizada

Podemos hacer un plot de la estructura que posee el objeto "neighborhood list", para visualizar usamos funciones descritas en Analyzing US Census.

```
# Filtramos para visualizar condado de Bronx
bronx <- nyc_data_prepped[str_detect(nyc_data_prepped$NAM, "Bronx"),]

# Guardamos su geometría (polygons)
bronx_geom <- st_geometry(bronx)

# Centroide de cada tramo censal y coordenadas
bronx_centroids = st_centroid(bronx_geom)
bronx_coordinates = st_coordinates(bronx_geom)

# Creamos neighbor list
bronx_nb_queen <- spdep::poly2nb(bronx)
bronx_nb_rook <- spdep::poly2nb(bronx, queen=FALSE)
```

Bronx Queen Model

```
bronx_nb_queen
```

```
## Neighbour list object:
## Number of regions: 242
## Number of nonzero links: 1066
## Percentage nonzero weights: 1.820231
## Average number of links: 4.404959
## 3 regions with no links:
## 86 90 242
```

Bronx Rook Model

```
bronx_nb_rook
```

```
## Neighbour list object:
## Number of regions: 242
## Number of nonzero links: 896
## Percentage nonzero weights: 1.52995
## Average number of links: 3.702479
## 4 regions with no links:
## 86 90 175 242
```

Para unir las dos imágenes en el mismo plot

```
# save plot
jpeg("image/bronx_neighborhood_rook_queen.jpg", width = 1000, height = 800)

# Preparamos plot de 2 imagenes
par(mfrow = c(1,2))

# Plot modelo Queen
plot(bronx_geom,
     main = "Queen",
     reset = FALSE,
     cex.main = 3)
plot(bronx_nb_queen, bronx_centroids,
     add = TRUE,
     col = 2,
     lwd = 1.5)

# Plot modelo Rook
plot(bronx_geom,
     main = "Rook",
     reset = FALSE,
     cex.main = 3)
plot(bronx_nb_rook, bronx_centroids,
     add = TRUE,
     col = 2,
     lwd = 1.5)

# close jpeg
dev.off()
```

Anexo 7: Códigos para implementación de modelo GRF y visualización de R2 local

Entrenamiento modelo GRF de SpatialML

El código empleado para usar el paquete *SpatialML* en el entrenamiento del modelo es el siguiente

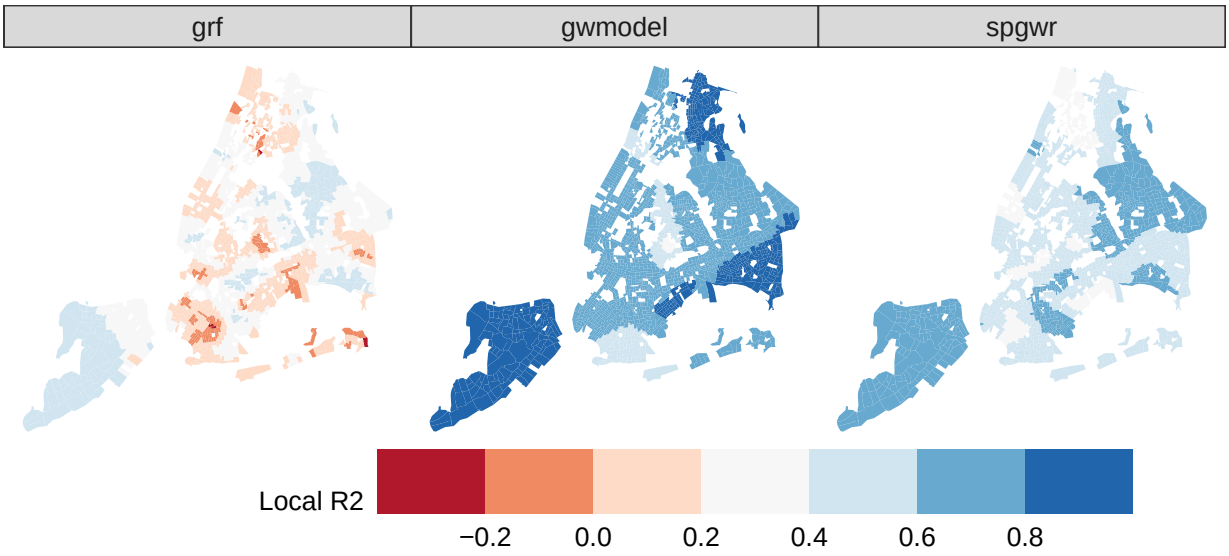
```
# Paquetes
library(SpatialML)
library(sf)
library(tidyverse)

# Obtenemos centroides
nyc_data_prepped <- nyc_data_prepped %>%
  mutate(lon = map_dbl(geometry, ~st_centroid(.x)[[1]]),
         lat = map_dbl(geometry, ~st_centroid(.x)[[2]]))

# Creamos columna de coordenadas
coords <- nyc_data_prepped %>%
  st_drop_geometry() %>%
  select(lat,lon)

# Filtramos
nyc_grf_prepped <- nyc_data_prepped %>%
  st_drop_geometry() %>%
  mutate(sqrt_med_value = sqrt(median_value)) %>%
  select(!c(GEOID, NAM, residuals, lagged_residuals, lat, lon))

# Definimos formula para GRF
formula_grf <- "sqrt_med_value ~ median_rooms + median_income +
pct_college + pct_foreign_born + pct_white + pct_black +
pct_hispanic + pct_asian + median_age + percent_ooh  +
pop_density"


# Optimizamos bandwitch
bwgrf <- grf.bw(formula = formula_grf,
                dataset = nyc_grf_prepped,
                kernel = "adaptive",
                coords = coords,
                bw.min = 98,
                bw.max = 98,
                step = 1,
                trees = 500,
                mtry = NULL,
                importance = "impurity",
                forests = FALSE,
                weighted = TRUE,
```

```
                verbose = TRUE)

# Entrena el modelo
grf_model <- grf(formula = formula_grf,
    dframe = nyc_grf_prepped,
    bw= 98,
    ntree = 500,
    mtry = 2,
    kernel = "adaptive",
    forests = TRUE,
    coords = coords)
```

Comparativa de R2 local entre modelos GWR y GRF

Usamos ggplot para visualizar los 3 modelos sobre el mismo mapa y bajo la misma escala de colores

```
library(ggthemes)
# pivot then plot
(ggplot_comp_loc_r2 <- nyc_data_prepped_sp %>%
  st_as_sf() %>%
  pivot_longer(cols = c("spgwr", "gwmodel", "grf")) %>%
  ggplot(aes(fill = value)) +
    geom_sf(color = NA) +
    scale_fill_fermenter(n.breaks = 8, palette = "RdBu", direction = 0) +
    ggthemes::theme_map(base_size = 8) +
    facet_wrap(~name) +
    labs(title = "Modelos GW: Comparativa R2 Local",
         fill = "Local R2") +
    theme(plot.title.position = 'plot',
          plot.title = element_text(hjust = 0.5,
                                    vjust = 3,
                                    size = 20),
          legend.position = c(0.20,-0.28),
          legend.key.height = unit(0.85, 'cm'),
          legend.key.width = unit(2, "cm"),
          legend.direction = "horizontal",
          legend.text = element_text(size=10),
          legend.title = element_text(size = 10),
          strip.text = element_text(size = 10)))
```




```
png('image/localR2GWM.png')
print(ggplot_comp_loc_r2)
dev.off()
```

```
## pdf
##   2
```