Spatial Machine Learning

Maximiliano S. Lioi

2023-01-12

Documentación

- Analyzing US Census Data - Kyle Walker
- Spatial Machine Learning - J. Morgan
- Import and Map NYC Census data into R with tidycensus

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 : Colleción 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.

Spatial Data

Viene representada en formas como:

- puntos (point reference data), i.e, ciudades en el mapa
- lineas (line string), i.e, caminos en el mapa
- poligonos (shapes) , i.e, distritos censales

Librerías

```
library(tidycensus)
library(tidyr)
library(censusapi)
library(tmap)
library(ggplot2)
library(dplyr)
library(stringr)
library(units)
library(stats)
library(grDevices)
library(dotenv)
library(sf)
```

Activacion API key

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")
```

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*

```
# 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
        )
```

Tidycensus, funciones principales para obtener data

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

De manera más general, 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.

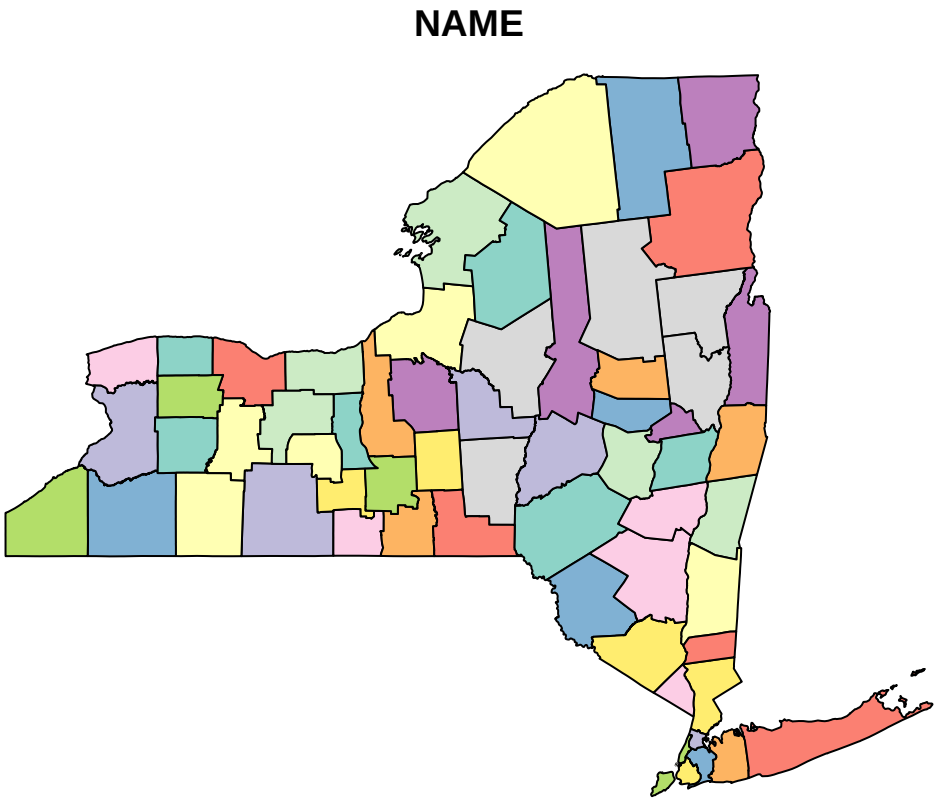
Para variables de la American Community Survey, debemos especificar el año de la encuesta, por ejemplo “acs1” para el primer año de la ACS, por ejemplo si se quiere acceder a la data *5-year ACS*

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

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"])
```

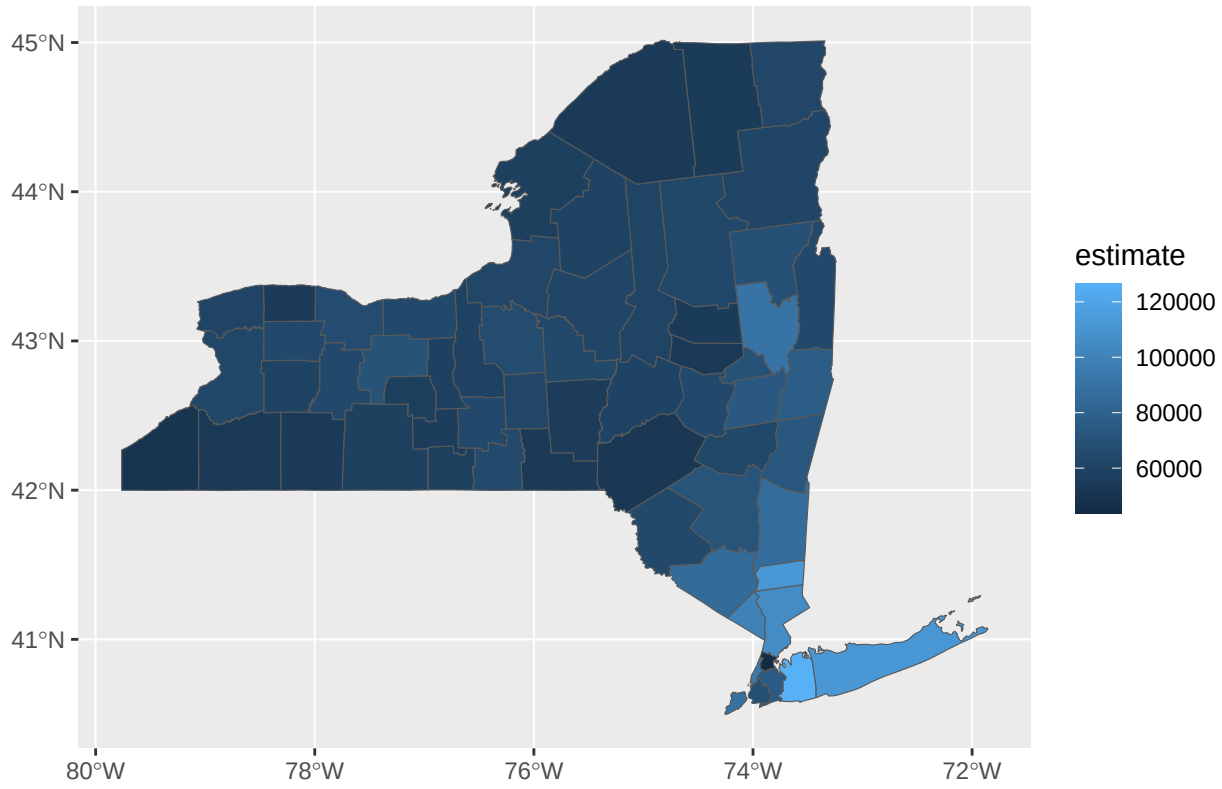


Plot del valor medio de las viviendas en New York

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".

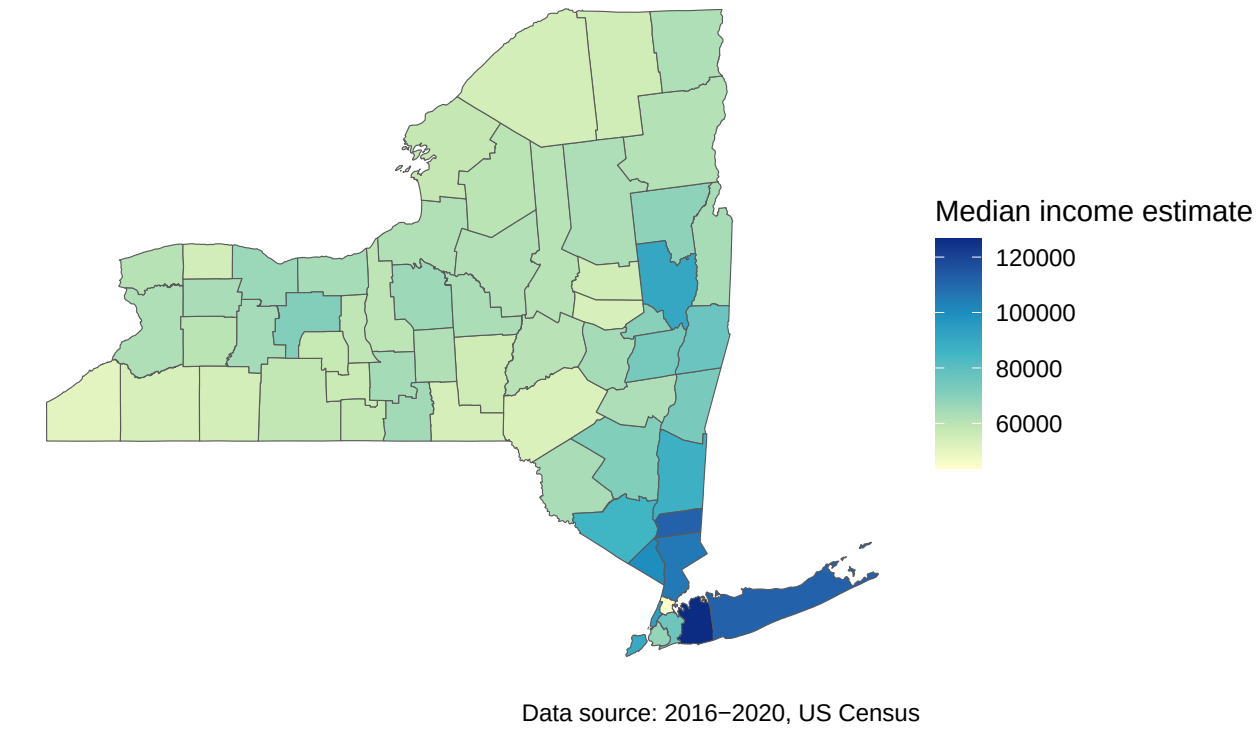
```
ggplot(data = medianincomenystate, aes(fill = estimate)) +
  geom_sf()
```



Customizing ggplot2 maps

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()
```

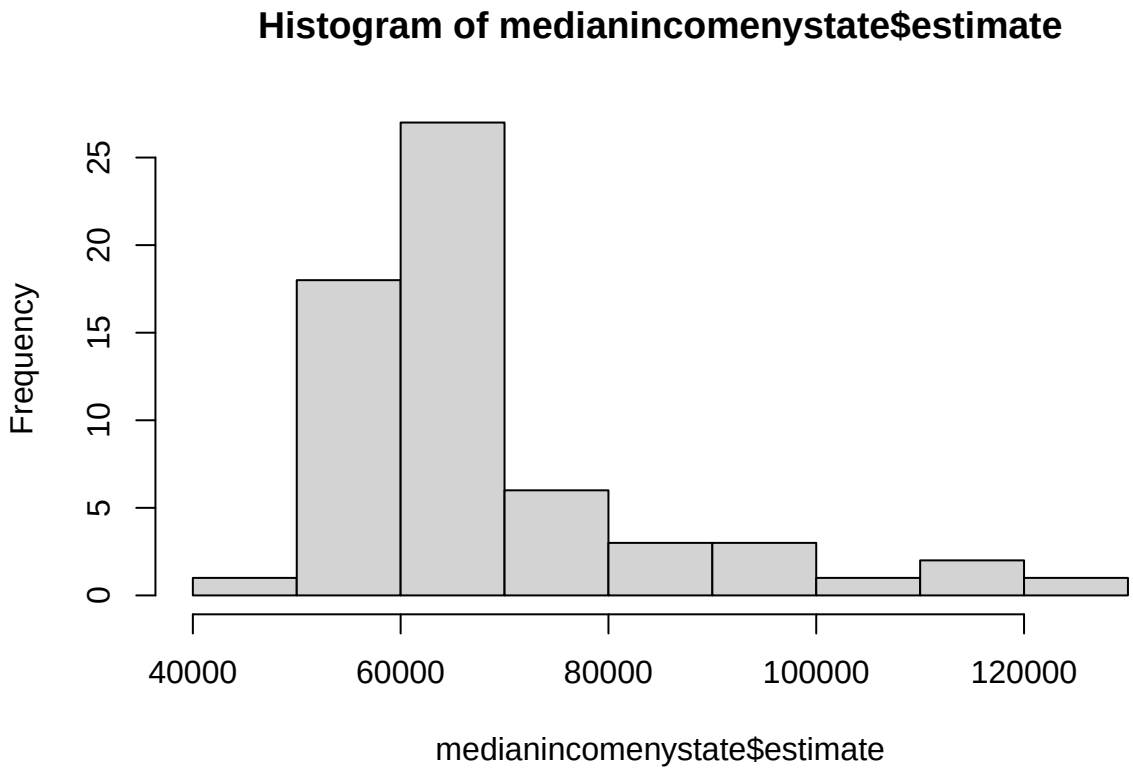


Las funciones que acompañan la sintaxis nos permiten customizar nuestro plot en ggplot2.

- 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

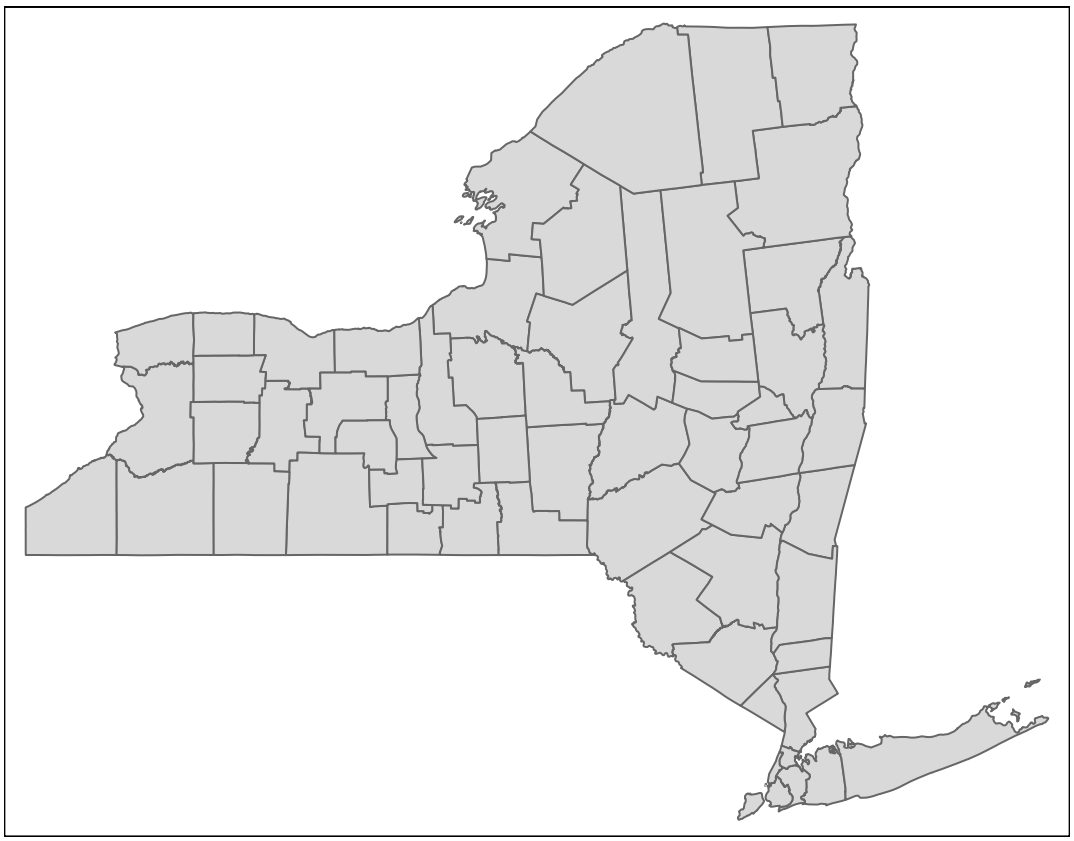
```
hist(medianincomenystate$estimate)
```



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()

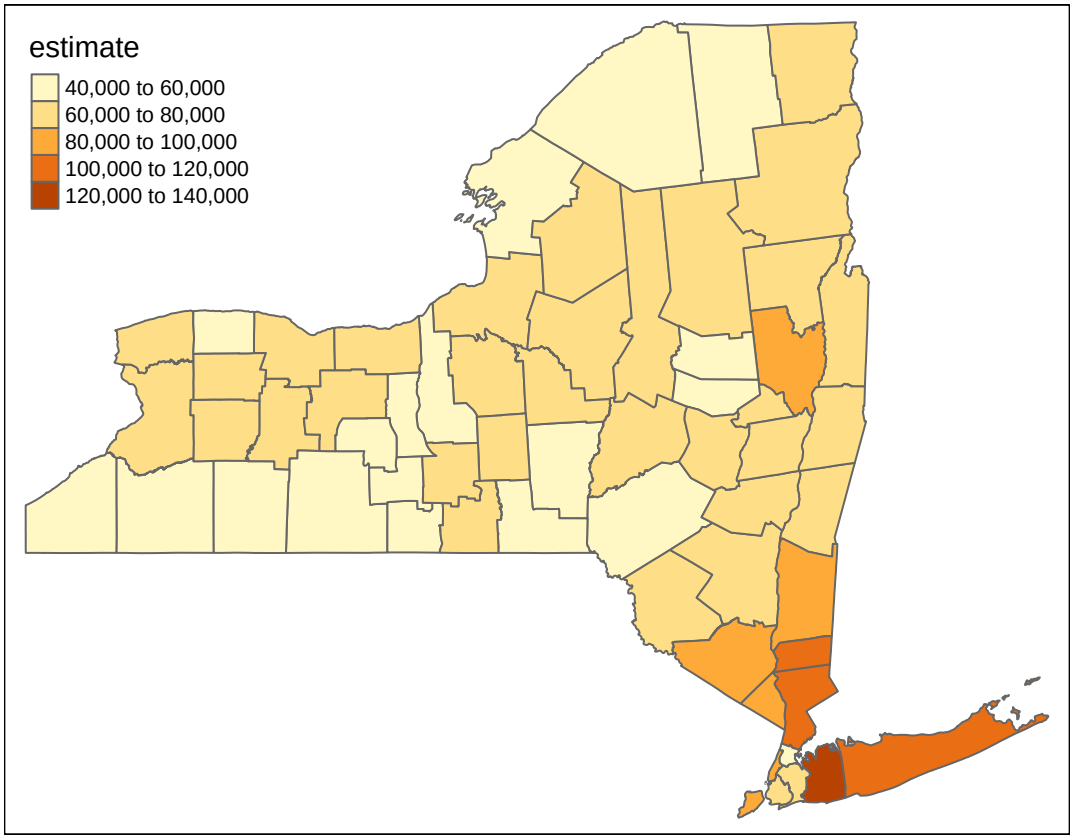
```
library(tmap)
tm_shape(medianincomenystate) +
  tm_polygons()
```



Variables en tmap

Veamos nuevamente la variable "median income"

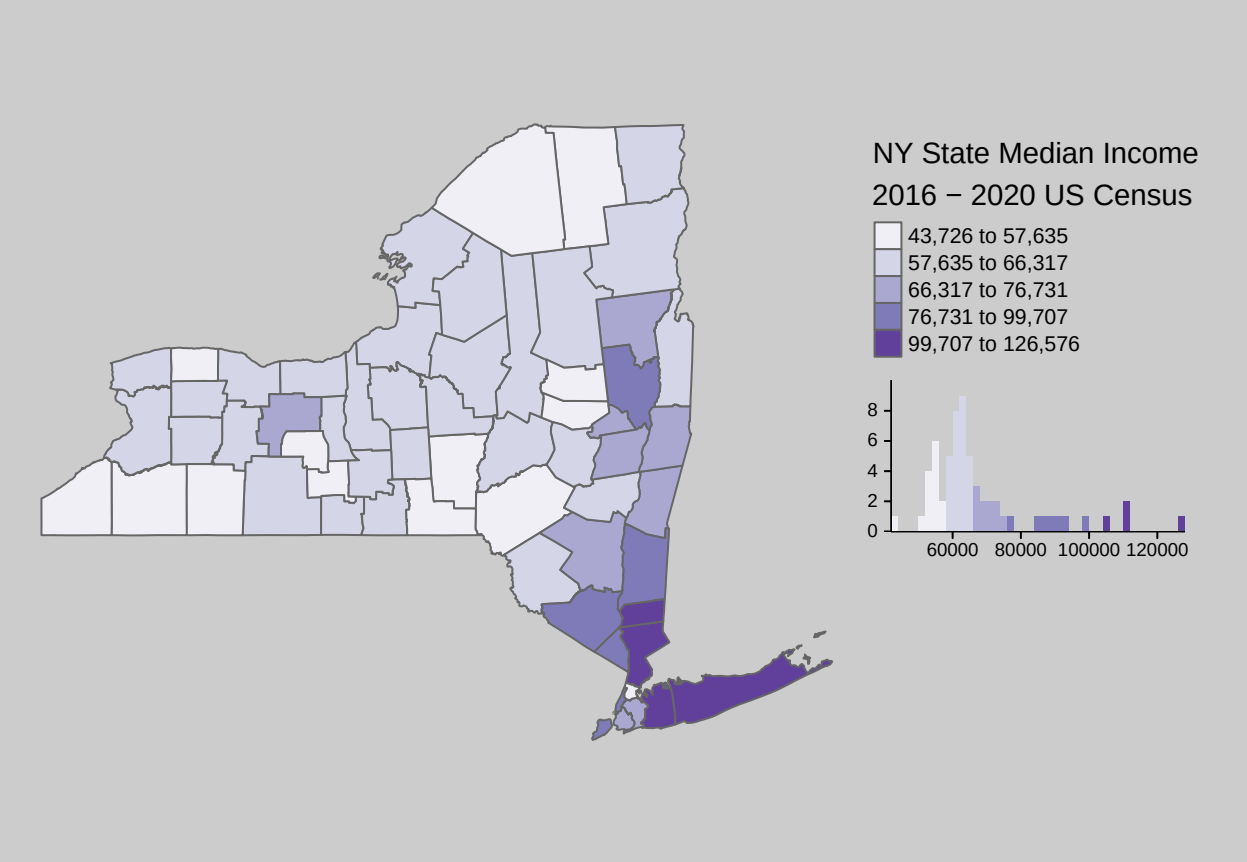
```
library(tmap)
tm_shape(medianincomenystate) +
  tm_polygons(col = "estimate")
```



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

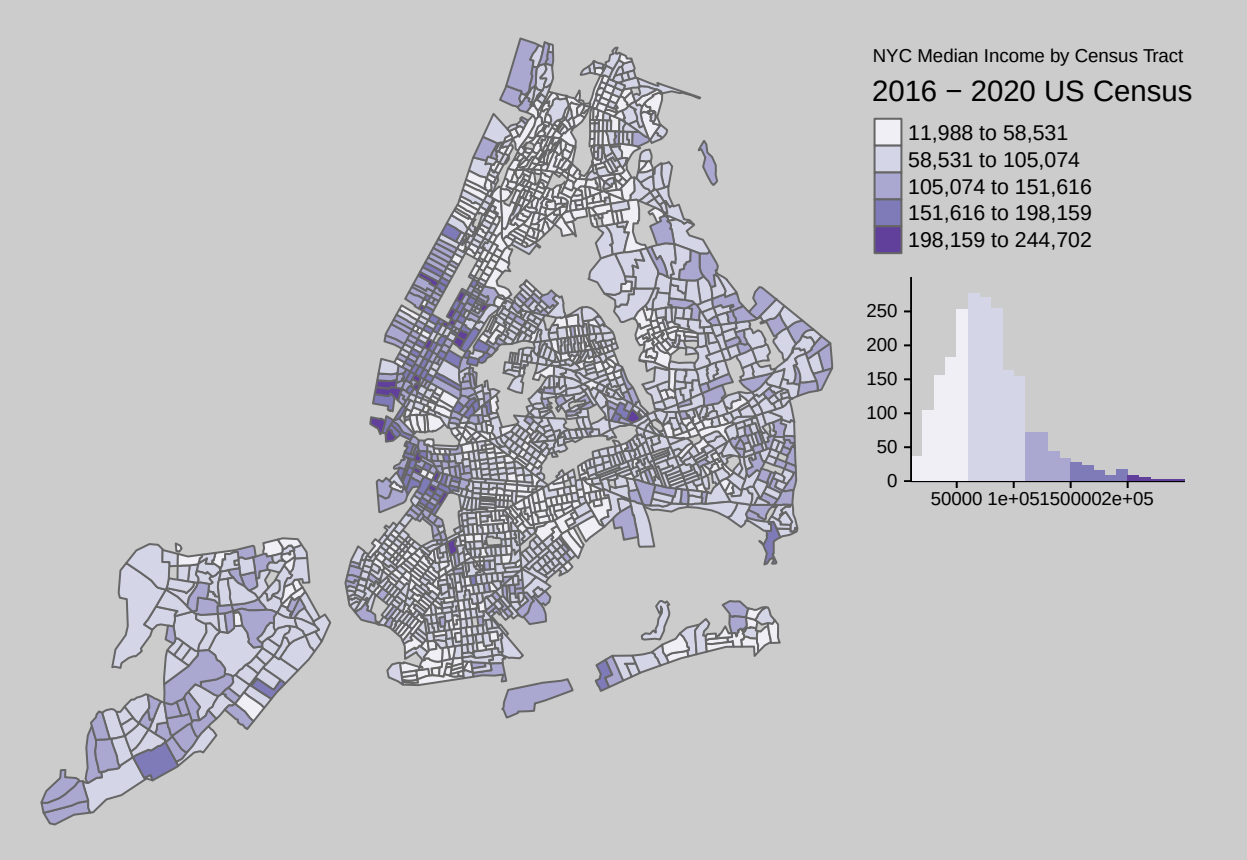
```
medianincomenyc <- separate(medianincomenyc,
         NAME,
         into = c("tract", "county"),
         sep = ", ")

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)
```



Spatial Machine Learning

Replicamos la simulación Spatial Machine Learning, Justin Morgan Williams, con el fin de entender los desafíos con los que se encuentra el Machine Learning con los datos espaciales, dicha simulación esta fuertemente basada en el capítulo 8 de Analyzing US Census Data, Modeling US Census Data, respecto de los desafíos al tratar datos espaciales, citando.

- *Autocorrelación espacial* → autocorrelación debida a la similitud en la ubicación del componente espacial de los datos
- *Heterogeneidad espacial* → datos que no siguen una distribución idéntica dentro del área de muestra
- *Limited Ground Truth* → muchas variables explicativas, verdad de terreno limitada
- *Multiple Scales and Resolutions* → puede existir en múltiples escalas y resoluciones

Si no se toman en cuenta, pueden tener efecto en la predicción de Machine Learning, entregando resultados que no son óptimos, la simulación trata los primeros 2 puntos, autoccorelación espacial y heterogeneridad espacial.

Importando data

Importamos data de NYC US Census con *tidycensus*

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

# set county variables, condados que forman NYC
nyc_counties <- c("Bronx","Kings","New York","Queens","Richmond")

# set list of census variables

# variable list
variables <- 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",
  median_year_built = "B25037_001",
  percent_ooh = "DP04_0046P"
)
```

Import usando get_acs()

Llamamos las variables antes definidas, en los condados que forman NYC usando get_acs()

```
# get acs data y transforma a tipo NYC EPSG
nyc_census_data <- get_acs(
  geography = "tract",
  variables = variables,
  state = "NY",
  county = nyc_counties,
  geometry = TRUE,
  output = "wide",
  year = 2020,
  key = Sys.getenv("CENSUS_API")) %>%
  st_transform(2263)
```

Las variables que importamos, que se corresponden a estimados que entrega la ACS sobre los condados de NY, son las siguientes:

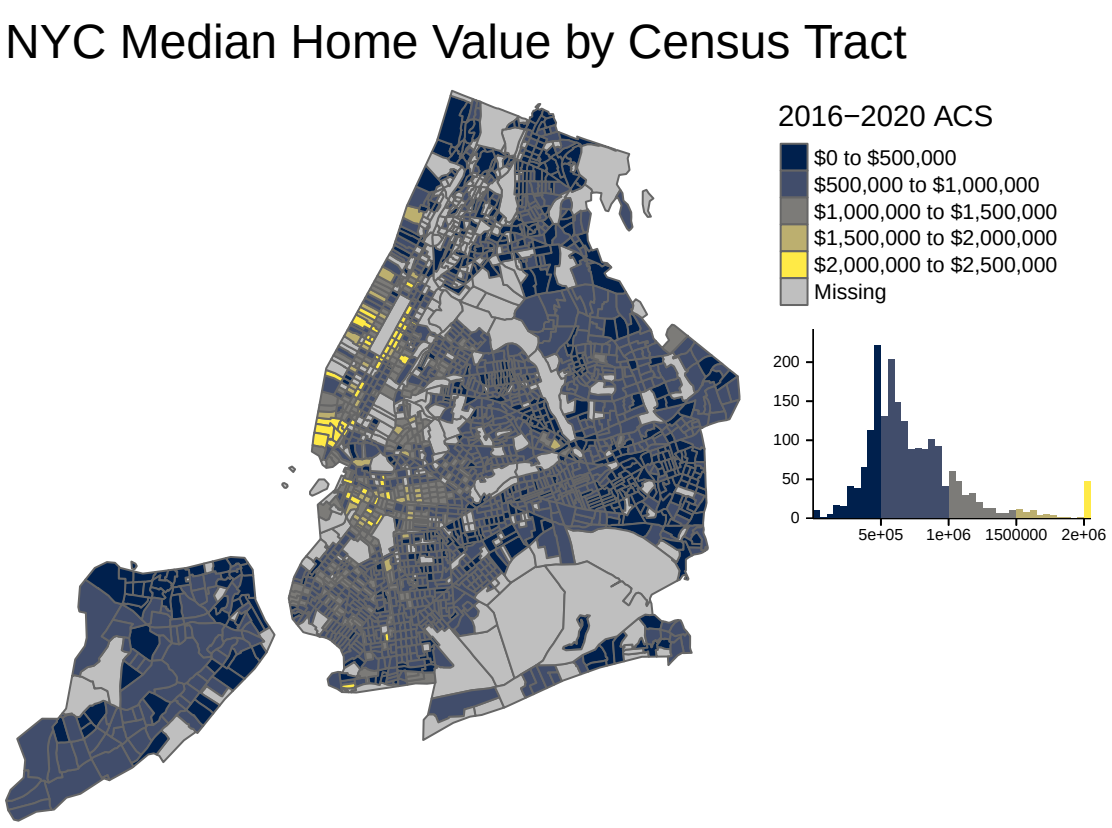
- median_valueE : El valor medio de la vivienda del tramo censal (nuestro outcome)
- median_roomsE : Cantidad media de habitaciones por casa en el tramo censal
- total_populationE : Población total
- median_ageE : Edad media de la población en el tramo censal
- median_year_builtE : Año promedio donde se construyó la vivienda
- median_incomeE : Ingreso medio de los hogares en el tramo censal
- pct_collegeE : Porcentaje de la población de 25 años o más con un título universitario de cuatro años
- pct_foreign_bornE: Porcentaje de la población que nació fuera de EE.UU
- pct_whiteE : Porcentaje de la población que se identifica como blanco no-hispano, se sigue la misma lógica con pct_blackE, pct_asian, pct_hispanic.
- percent_oohE : El porcentaje de unidades de vivienda en el tramo censal que están ocupadas por sus propietarios.

Más detalles sobre la estructura del código se encuentran en la sección 8.2.1 : Data setup and exploratory data analysis}, en el libro se toma la misma variable “median home value”, pero que a diferencia de esta réplica, 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

Plot de la variable dependiente

Nuestra variable dependiente será el valor promedio de la vivienda en NYC.

```
# plot tmap
(nyc_median_value_hist_tm <- nyc_census_data[!st_is_empty(nyc_census_data),,drop=F] %>%
tm_shape() +
  tm_polygons(col = "median_valueE",
          palette = "cividis",
          title = "2016-2020 ACS",
          legend.hist = TRUE,
          legend.format = scales::dollar_format()) +
  tm_layout(main.title = "NYC Median Home Value by Census Tract",
            frame = FALSE,
            legend.outside = TRUE,
            bg.color = "grey100",
            legend.hist.width = 5,
            ))
```



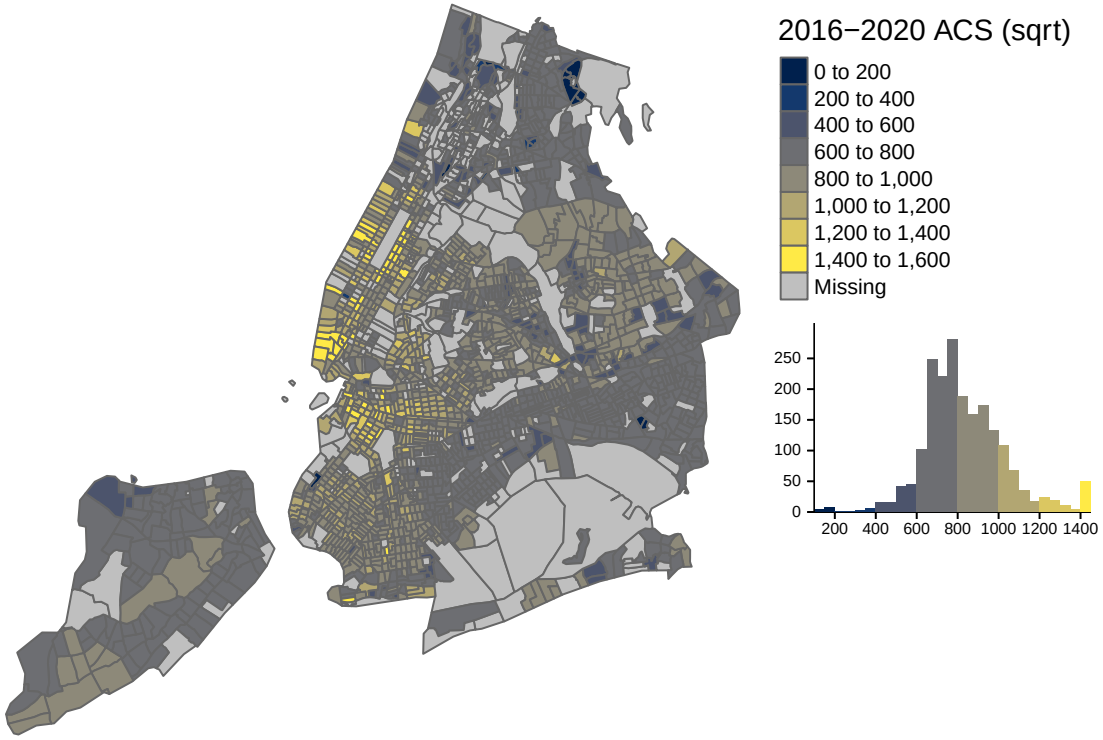

```
# save plot
tmap_save(nyc_median_value_hist_tm, "nyc_median_value.png", width=1920, height=1080, asp=0)
```

Notamos en el histograma que se presenta una asimetría a la derecha, existe una población no menor cuyos hogares perciben ingresos mucho mayores a la mediana, y vemos que existe data perdida, para el primer punto, podemos aplicar una transformación a la variable dependiente, tomando su raiz cuadrada para reducir esta asimetría a la derecha y tener los datos mejor distribuidos, acercándose más a una distribución normal, y en consecuencia, entregando resultados más precisos.

Transformación de los datos

Creamos la nueva variable tomando la raiz cuadrada de la variable dependiente median_valueE

```
# create plot
(nyc_median_value_sqrt_hist_tm <- nyc_census_data[!st_is_empty(nyc_census_data),,drop=F] %>% #
   mutate(sqrt_med_value = sqrt(median_valueE)) %>% #variable_sqrt
tm_shape() +
  tm_polygons(col = "sqrt_med_value",
          palette = "cividis",
          title = "2016-2020 ACS (sqrt)",
          legend.hist = TRUE) +
  tm_layout(main.title = "NYC Median Home Value by Census Tract",
            frame = FALSE,
            legend.outside = TRUE,
            bg.color = "grey100",
            legend.hist.width = 5,
            ))
```




```
# save plot
tmap_save(nyc_median_value_sqrt_hist_tm, "nyc_median_value_sqrt.png", width=1920, height=1080,
```

Preparando la data para modelar

Debemos borrar aquellas variables con data NA, y eliminar columnas con información del margen de error eliminando las columnas que terminan con "M", se le quita la E final a las columnas como median_valueE y se añaden las variables pop_density y median_structure_age, esto pues buscamos transformar los predictores de manera que se represente mejor la relación que tiene con la variable de salida (en este caso, median_value)

- pop_density → mide densidad de población por metros cuadrados
- median_structure_age → resta 2020 de median_year_built

```
# load packages
#library(dplyr)
#library(stringr)
#library(units)
#library(stats)

# prep data for model
nyc_census_data_prepped <- nyc_census_data %>%
  mutate(pop_density = as.numeric(set_units(total_populationE / st_area(.),
 "1/km2")),
         median_structure_age = 2020 - median_year_builtE) %>%
  select(!ends_with("M")) %>% # drop margin of error cols
  rename_with(.fn = ~str_remove(.x, "E$")) %>% # remove E from col name
  na.omit() # omit NA

nyc_census_data_prepped
```

```
## Simple feature collection with 1967 features and 17 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 913037.2 ymin: 120117 xmax: 1067245 ymax: 272608.6
## Projected CRS: NAD83 / New York Long Island (ftUS)
## # A tibble: 1,967 x 18
##    GEOID   NAM   media~1 media~2 total~3 media~4 media~5 media~6 pct_c~7 pct_f~8
##    <chr>   <chr>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
##  1 360050~ Cens~  426200     5.1    4167    37.9    1962   68000    34.3    27
##  2 360050~ Cens~  437600     5      5684    40.1    2002   93155    28.9    18.6
##  3 360050~ Cens~  575200     4.1    5917    39.3    1973   34766    16.6    24.3
##  4 360050~ Cens~  350000     3.9    1334    30.9       0   53882    17.1    24.3
##  5 360050~ Cens~  447100     3.9    4768    38.4    1945   48711    17.1    38.2
##  6 360050~ Cens~  653800     3.7    5694    32      1953   27196     8.8    38.4
##  7 360050~ Cens~  289800     4.2    5030    43.7    1967   52068    34.5     8.4
##  8 360050~ Cens~  242300     4      1858    30.5    1945   42228    19.4    28.3
##  9 360050~ Cens~  476400     3.8    3140    33.3    1980   22159    10.4    23.9
## 10 360050~ Cens~  594900     3.7    3738    26.6    1965   29068    13.5    23.6
```

```
## # ... with 1,957 more rows, 8 more variables: pct_white <dbl>, pct_black <dbl>,
## #   pct_hispanic <dbl>, pct_asian <dbl>, percent_ooh <dbl>,
## #   geometry <MULTIPOLYGON [US_survey_foot]>, pop_density <dbl>,
## #   median_structure_age <dbl>, and abbreviated variable names 1: median_value,
## #   2: median_rooms, 3: total_population, 4: median_age, 5: median_year_built,
## #   6: median_income, 7: pct_college, 8: pct_foreign_born
```

Modelos

En esta sección pasaremos al desarrollo de modelos de predicción, para ello replicamos los modelos de predicción hecho en Spatial Machine Learning, Justin Morgan Williams y además, complementamos con el Capítulo 8 del libro 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. Nos enfocamos en particular en la segunda sección del capítulo.

Simple Linear Regression

Con la data preparada, podemos crear el primer modelo sencillo de regresión lineal, con la variable dependiente, la raiz cuadrada del valor medio de la vivienda ("NYC Median Home Value")

```
# model formula
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 + median_structure_age + pop_density"

# compute model
model1 <- lm(formula = formula, data = nyc_census_data_prepped)

# view model statistics
summary(model1)
```

```
##
## Call:
## lm(formula = formula, data = nyc_census_data_prepped)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -637.50  -80.48   -1.49   75.05  707.21
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)           5.041e+02  3.903e+01  12.914  < 2e-16 ***
## median_rooms          5.331e+01  6.420e+00   8.304  < 2e-16 ***
## median_income         2.058e-03  1.672e-04  12.310  < 2e-16 ***
## pct_college           1.443e+00  3.565e-01   4.046 5.41e-05 ***
## pct_foreign_born     -9.391e-01  3.203e-01  -2.932  0.00341 **
## pct_white             2.371e+00  2.332e-01  10.169  < 2e-16 ***
## pct_black             1.060e+00  2.035e-01   5.211 2.08e-07 ***
## pct_hispanic         -1.086e-02  2.116e-03  -5.133 3.14e-07 ***
```

```
## pct_asian             3.281e+00  2.947e-01  11.132  < 2e-16 ***
## median_age           -1.824e+00  5.882e-01  -3.101  0.00195 **
## percent_ooh          -4.140e+00  2.785e-01 -14.866  < 2e-16 ***
## median_structure_age  3.234e-02  3.569e-03   9.061  < 2e-16 ***
## pop_density           7.144e-04  3.355e-04   2.129  0.03337 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 143.9 on 1954 degrees of freedom
## Multiple R-squared:  0.4789, Adjusted R-squared:  0.4757
## F-statistic: 149.6 on 12 and 1954 DF,  p-value: < 2.2e-16
```

Aquellas variables con mayor p-value son pct_foreign_born, median_age, pop_density, un p valor alto nos dice que la variable no tiene mucha significancia en el resultado (típicamente pedimos que sea menor a 0.05 para que la variable se considere significativa), 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, notamos que el valor medio de la vivienta aumenta, podemos ver tambien el valor R^2 el cual nos dice que un 0.4757% de la varianza de median_value es explicado con las variables del modelo (el modelo es deficiente).

En la regresión lineal, los errores no son independientes en un modelo con componentes espaciales, esto es porque la autocorrelación espacial está presente en el error, lo que nos dice que el performance del modelo depende de la posición geográfica.

Para saber si este es el fenómeno que se presenta, tomamos en cuenta el Índice de Moran.

Índice de Moran

Estudiamos (Jiang 2019)

$$I = \frac{N}{W} \sum_{i,j} w_{ij}(x_i - \bar{x})(x_j - \bar{x})$$

Spatial Weights Matrix

```
# create neighbor list object
nyc_nb <- spdep::poly2nb(nyc_census_data_prepped, queen = TRUE)
```

Referencias

Jiang, Zhe. 2019. "A Survey on Spatial Prediction Methods." *IEEE Transactions on Knowledge and Data Engineering* 31 (9): 1645–64. https://doi.org/10.1109/TKDE.2018.2866809.