Spatial Machine Learning

Maximiliano S. Lioi

2023-01-12

Documentación

- Analyzing US Census Data - Kyle Walker
- Spatial Machine Learning
- 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)
```

```
##
## Attaching package: 'censusapi'

## The following object is masked from 'package:methods':
##
##     getFunction
```

```
library(tmap)
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(stringr)
library(units)
```

```
## udunits database from C:/Users/Maximiliano/AppData/Local/R/win-library/4.2/units/share/udunits/uduni
```

```
library(stats)
```

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
        )
```

```
## Getting data from the 2017-2021 5-year ACS
```

```
## Downloading feature geometry from the Census website.  To cache shapefiles for use in future sessions
```

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.

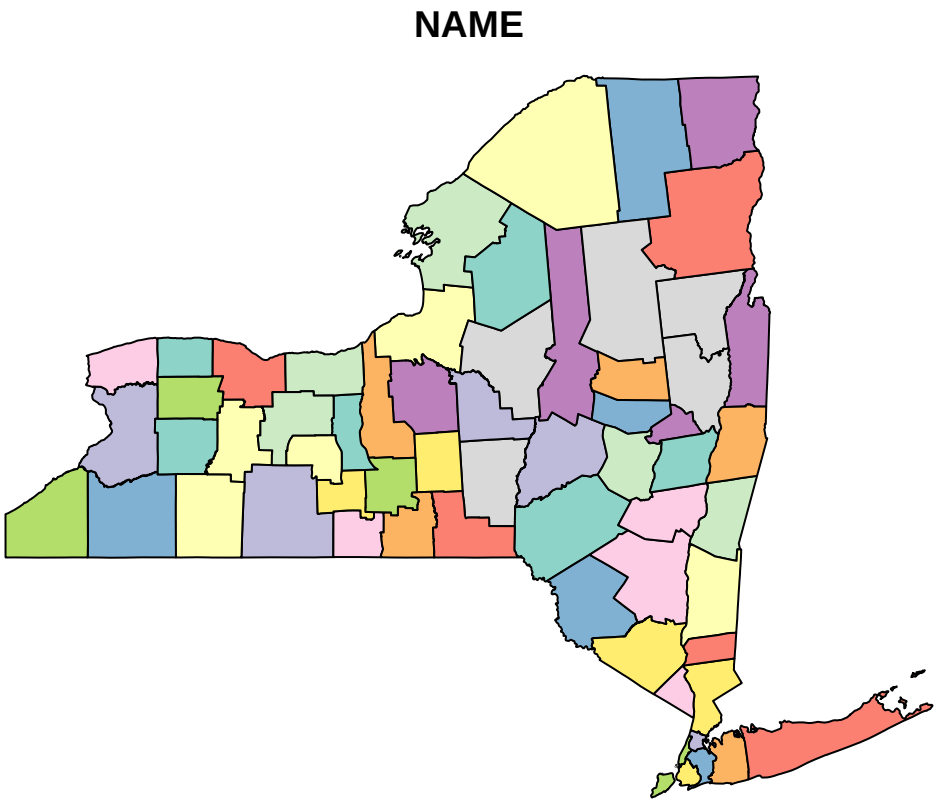
Para variables de la American Community Survey, debemos especificar el año de la encuesta, por ejemplo “acs1” para el primer año de la ACS, por ejemplo si se quiere acceder a la data *5-year ACS*

```
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"])
```

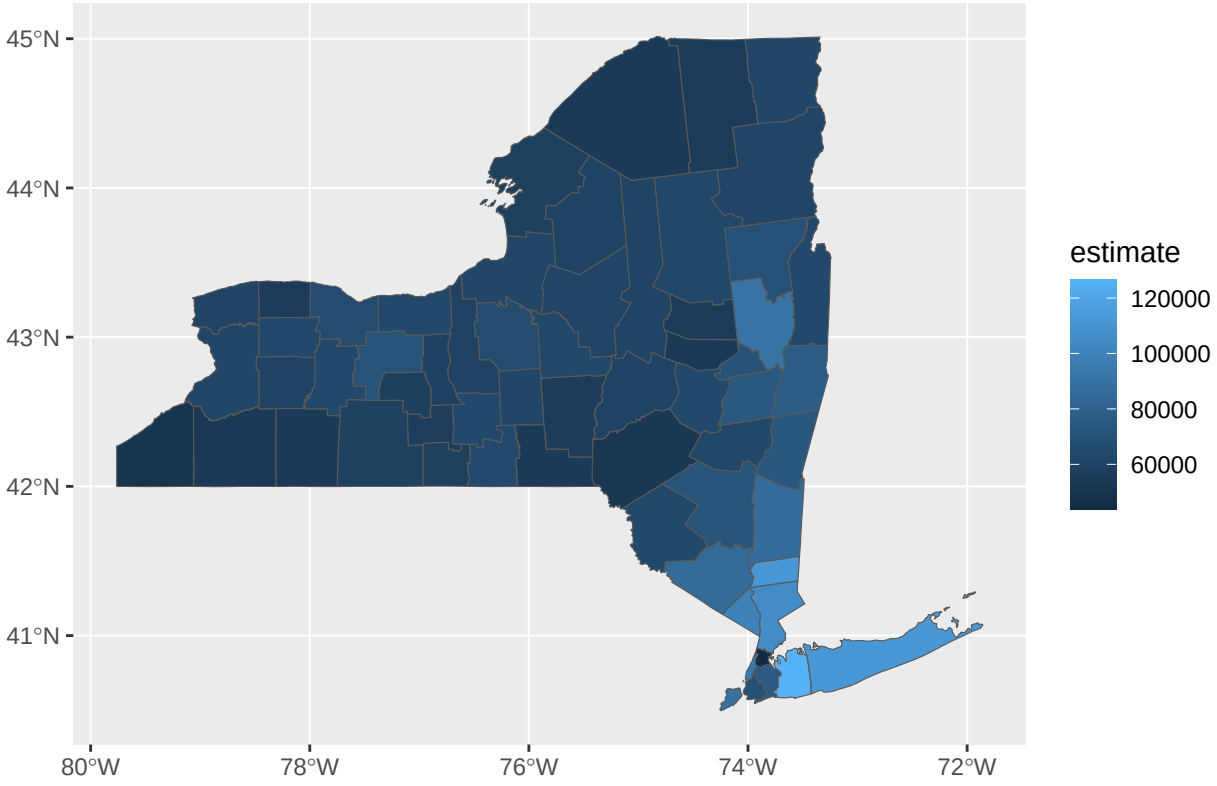


Plot del valor medio de las viviendas en New York

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".

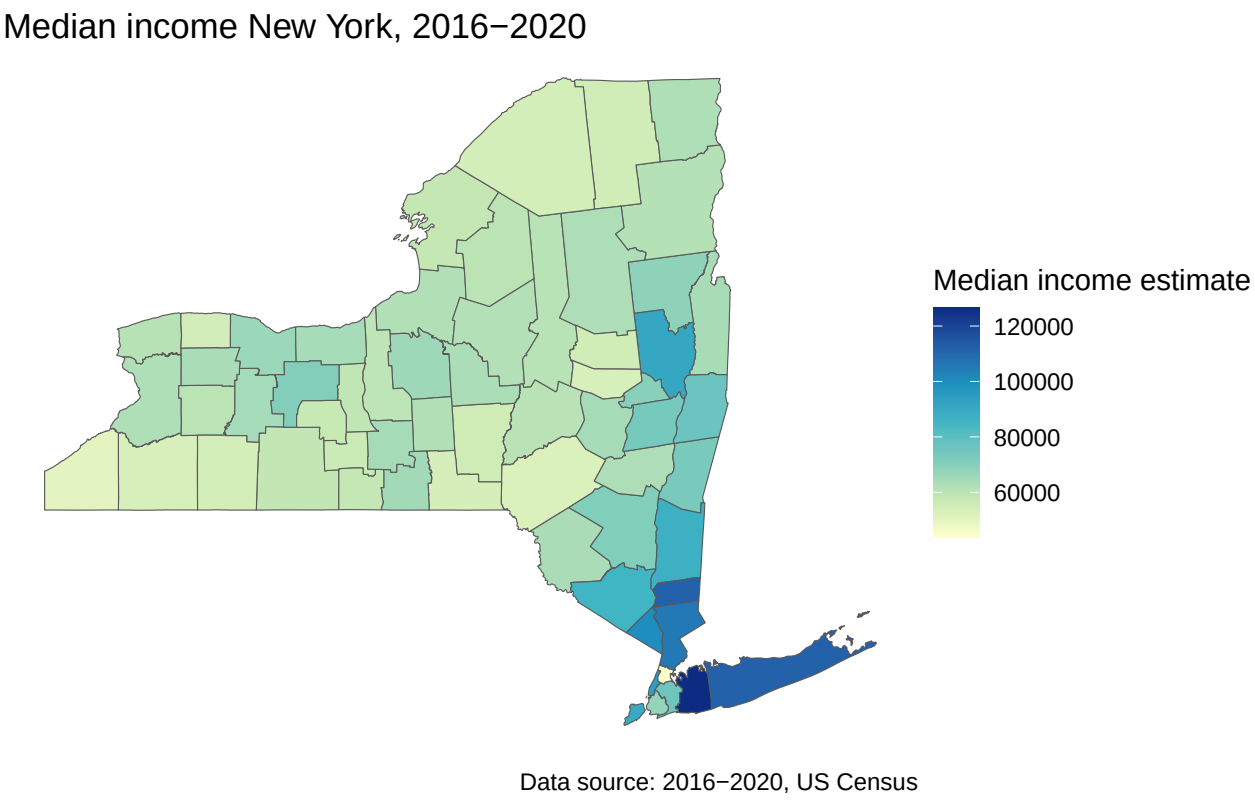
```
ggplot(data = medianincomenystate, aes(fill = estimate)) +
  geom_sf()
```



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()
```

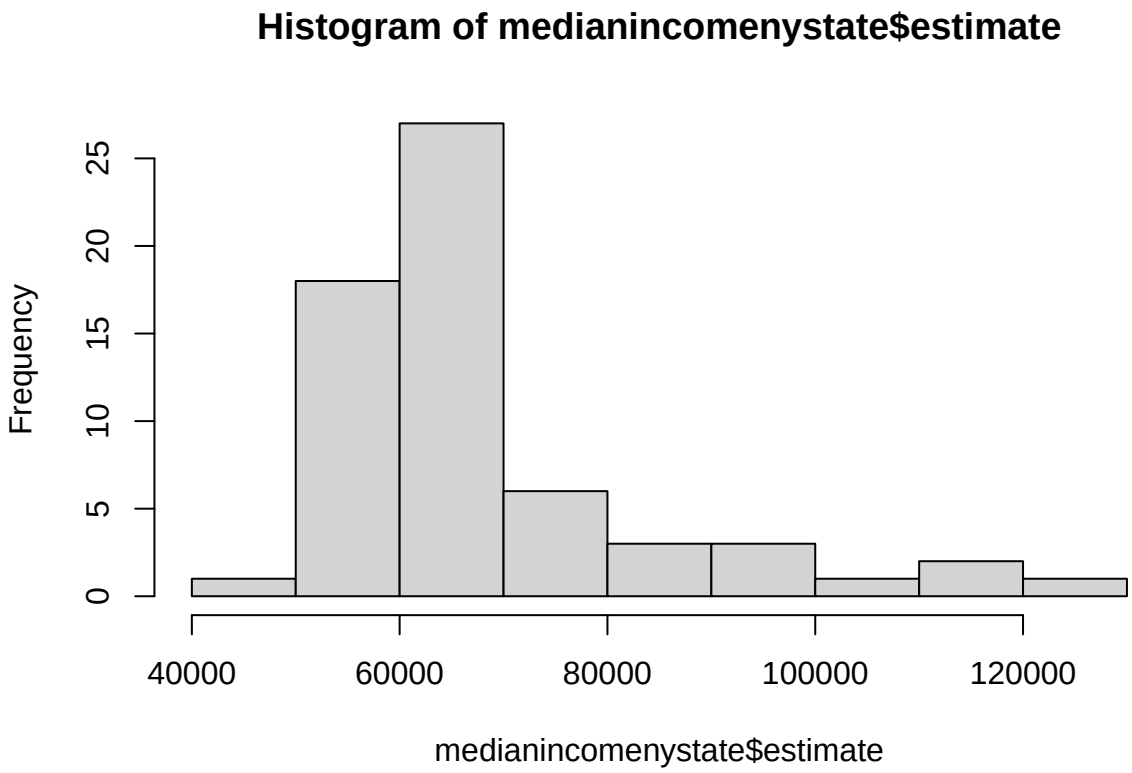


Las funciones que acompañan la sintaxis nos permiten customizar nuestro plot en ggplot2.

- 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

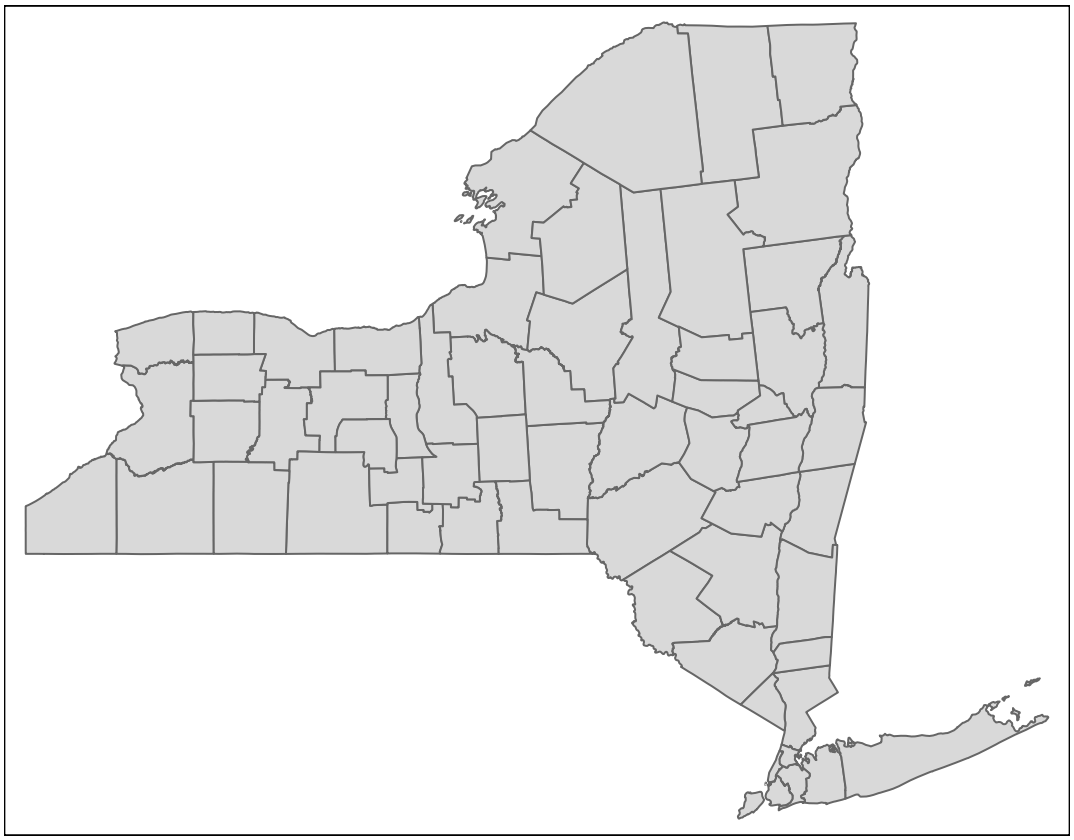
```
hist(medianincomenystate$estimate)
```



Map-making con tmap

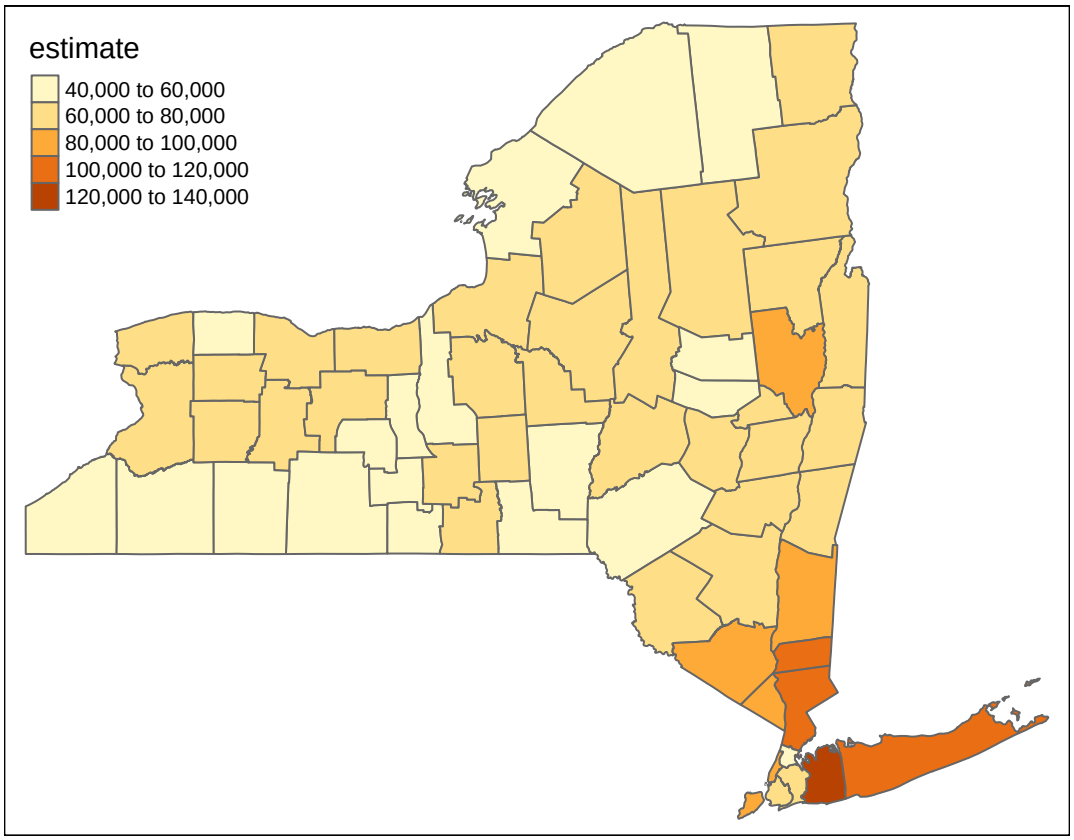
La sintaxis en similar a la usada en *ggplot2*, el objeto mapa se inicializa con la función tm_shape() y nos permite visualizar los distritos censales con tm_polygons()

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



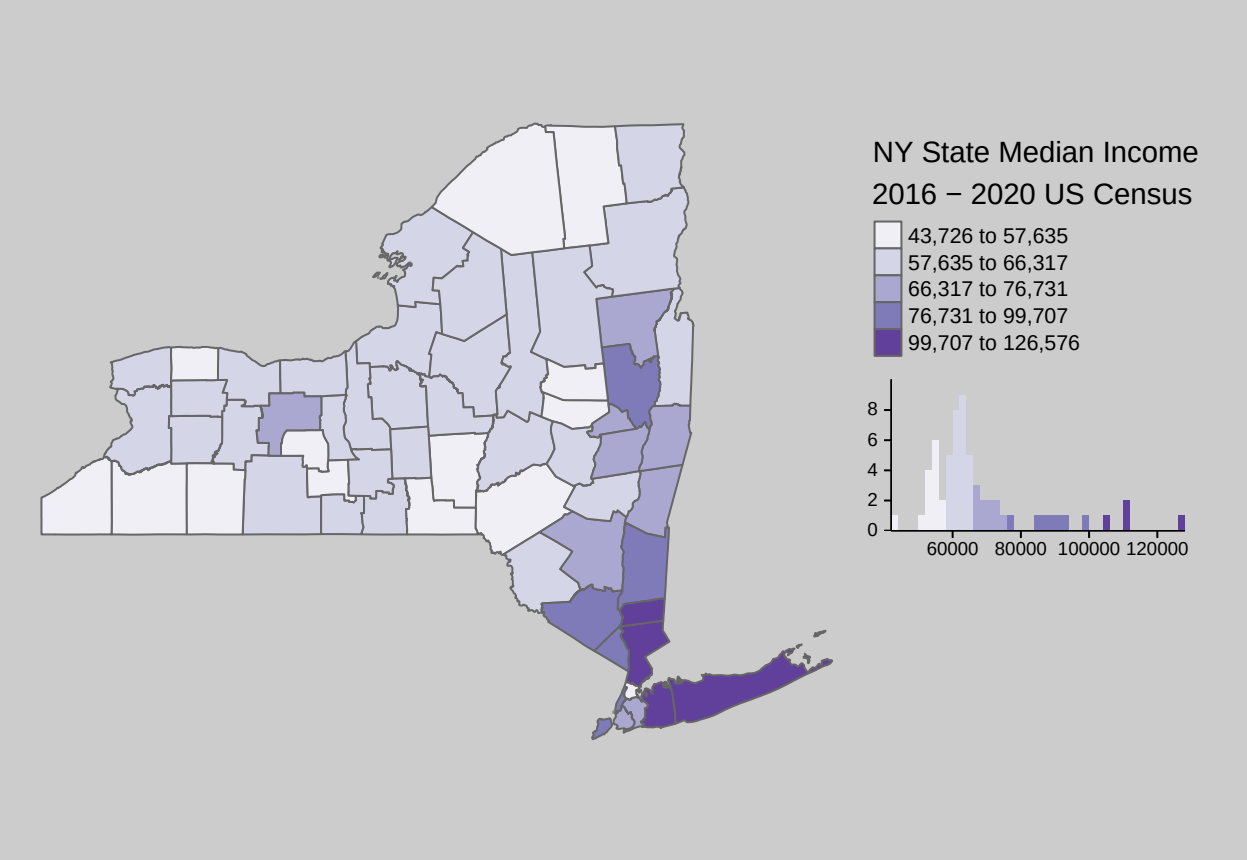
Veamos nuevamente la variable “median income”

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



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
        )
```

```
## Getting data from the 2017-2021 5-year ACS
```

```
## Downloading feature geometry from the Census website.  To cache shapefiles for use in future sessions
```

```
##   |                                                                              |
```

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

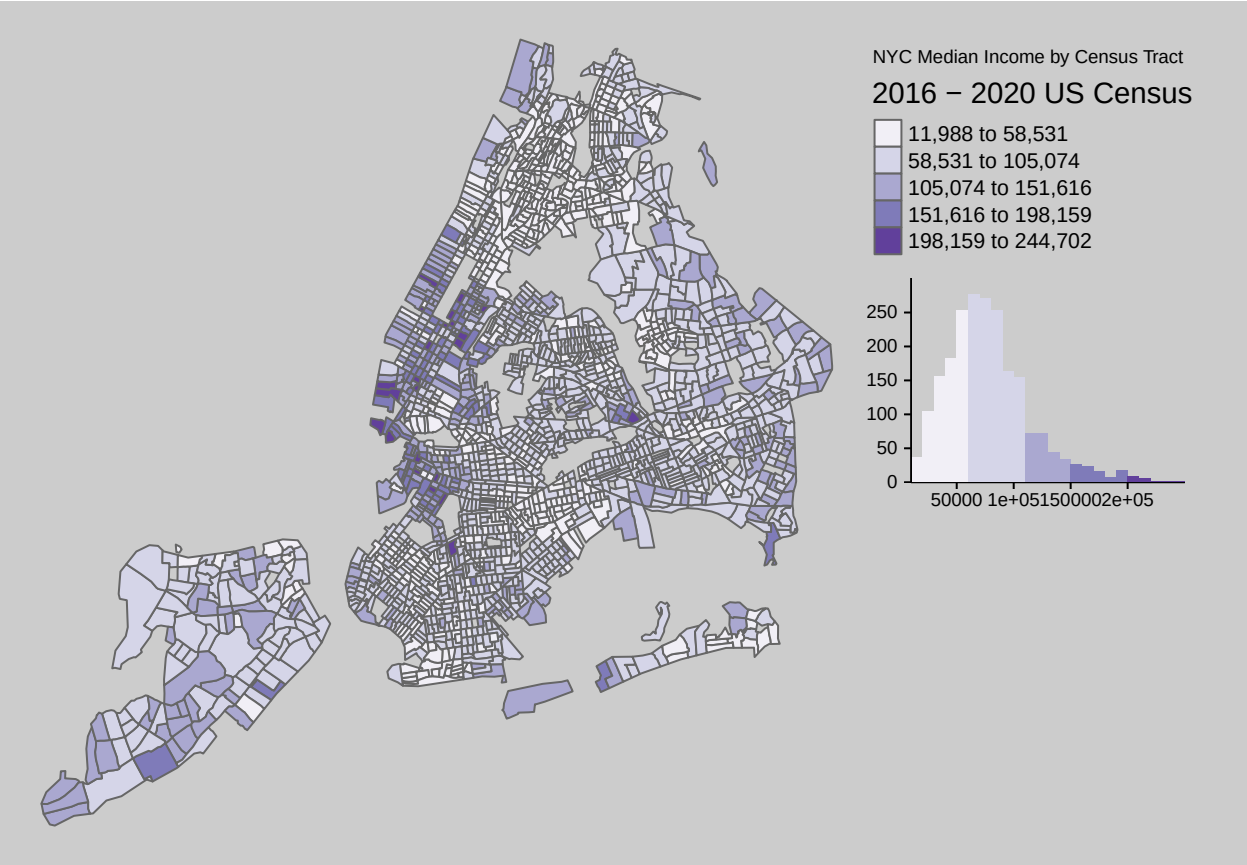
```
medianincomenyc <- separate(medianincomenyc,
         NAME,
         into = c("tract", "county"),
         sep = ", ")
```

```
## Warning: Expected 2 pieces. Additional pieces discarded in 5411 rows [1, 2, 3,
## 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, ...].
```

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

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

```
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)
```



Indice de Moran

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