

Proyecto DataViz

Jorge Arteaga y Adriana Palacio

2021-12-05

Contents

1 Sobre este libro	5
1.1 Objetivos	5
1.2 Paquetes Necesarios	5
2 Cargue y Limpieza de Datos	7
2.1 Carga de datos	7
2.2 Revisión de datos faltantes	8
2.3 Creación en base de datos	13
3 Análisis Exploratorio de los datos	15
Resumen de los datos	15
Datos Atípicos	17
Análisis del anfitrión	23
Análisis de los precios	25
Visualización en mapas	28

Chapter 1

Sobre este libro

Este libro contiene el detalle del proyecto enfocado a sistemas de información geográfica. Se estará trabajando con un archivo tomado de *Kaggle* en <https://www.kaggle.com/dgomonov/data-exploration-on-nyc-airbnb> que contiene información resumida y métricas para Airbnb en la ciudad de Nueva York en 2019.

1.1 Objetivos

a donde se desea llegar con todo el análisis presentado en la entrega final les pido el favor que agreguen imágenes a los books, como por ejemplo logo de la universidad, así como también cualquier imagen relacionada con el proyecto, que hagan visualmente más atractiva la portada del book.

1.2 Paquetes Necesarios

Para poder trabajar con datos se hace necesario cargar una serie de librerías.

1. Para el cargue del archivo csv, utilizaremos el paquete `readr`.
2. Para el manejo de dataframe, utilizaremos el paquete `dplyr`.
3. Para revisión de datos faltantes, utilizaremos los paquetes `mice` y `VIM`.
4. Para conexión a la base de datos PostgreSQL, utilizaremos el paquete `RPostgresL`.
5. Para poder hacer uso de la API y realizar la conexión a la base de datos con éxito, utilizaremos el paquete `DBI`.
6. Para graficar, utilizararemos el paquete `ggplot2`
7. Para visualización en mapas, utilizaremos el paquete `ggmap`

```
library(readr)
library(dplyr)
```

```
library(mice)
library(VIM)
library(DBI)
library(RPostgres)
library(ggplot2)
library(ggmap)
```

Chapter 2

Cargue y Limpieza de Datos

En este capítulo abordaremos el cargue de los datos de AIRBNB de la ciudad de Nueva York en el año 2019.

2.1 Carga de datos

El archivo CSV con el listado de Airbnb de la ciudad de Nueva York para el año 2019 descargado de *Kaggle* se cargará en una base de datos en Heroku Postgress. Pero para lograr esto, primero debemos cargar como `dataframe` el archivo a través de la función `read.csv()`, agregando la instrucción `na = c("", "NA")` para tomar los valores vacíos como datos faltantes `na`.

```
airbnb <- read.csv(file = "Datasets/AB_NYC_2019.csv", na = c("", "NA"))
head(airbnb)
```

```
##      id                               name host_id  host_name
## 1 2539        Clean & quiet apt home by the park     2787       John
## 2 2595           Skylit Midtown Castle     2845    Jennifer
## 3 3647 THE VILLAGE OF HARLEM....NEW YORK !     4632 Elisabeth
## 4 3831            Cozy Entire Floor of Brownstone     4869 LisaRoxanne
## 5 5022 Entire Apt: Spacious Studio/Loft by central park     7192      Laura
## 6 5099 Large Cozy 1 BR Apartment In Midtown East     7322      Chris
##   neighbourhood_group neighbourhood latitude longitude      room_type price
## 1             Brooklyn      Kensington 40.64749 -73.97237 Private room  149
## 2            Manhattan        Midtown 40.75362 -73.98377 Entire home/apt  225
## 3            Manhattan         Harlem 40.80902 -73.94190 Private room  150
## 4             Brooklyn     Clinton Hill 40.68514 -73.95976 Entire home/apt   89
## 5            Manhattan    East Harlem 40.79851 -73.94399 Entire home/apt   80
## 6            Manhattan     Murray Hill 40.74767 -73.97500 Entire home/apt  200
##   minimum_nights number_of_reviews last_review reviews_per_month
```

```

## 1           1           9 2018-10-19      0.21
## 2           1          45 2019-05-21      0.38
## 3           3           0 <NA>          NA
## 4           1          270 2019-07-05     4.64
## 5          10           9 2018-11-19      0.10
## 6           3          74 2019-06-22     0.59
##   calculated_host_listings_count availability_365
## 1                               6            365
## 2                               2            355
## 3                               1            365
## 4                               1            194
## 5                               1              0
## 6                               1            129

```

Este archivo contiene 48.895 registros y 16 variables para análisis. En la siguiente sección revisaremos si existen datos faltantes en el dataset.

2.2 Revisión de datos faltantes

Para determinar la existencia de datos faltantes en el dataframe *Airbnb*, primero determinaremos por columna cual es su proporción de datos faltantes contando los valores na.

```
pMiss <- function(x){sum(is.na(x))/length(x)*100}
apply(airbnb,2,pMiss)
```

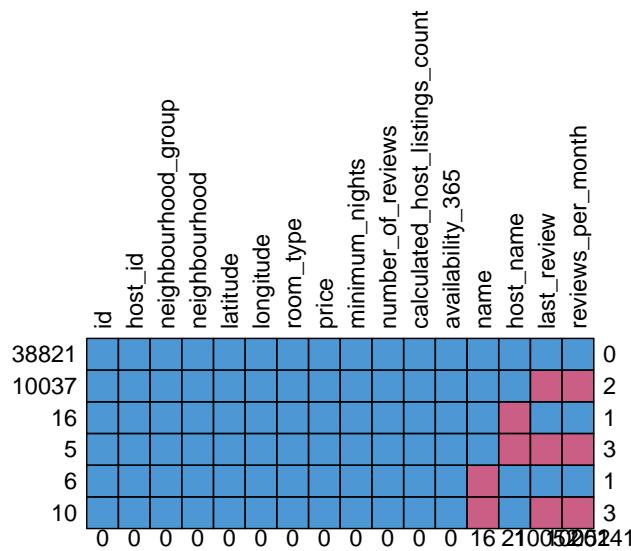
	id	name
##	0.00000000	0.03272318
##	host_id	host_name
##	0.00000000	0.04294918
##	neighbourhood_group	neighbourhood
##	0.00000000	0.00000000
##	latitude	longitude
##	0.00000000	0.00000000
##	room_type	price
##	0.00000000	0.00000000
##	minimum_nights	number_of_reviews
##	0.00000000	0.00000000
##	last_review	reviews_per_month
##	20.55833930	20.55833930
##	calculated_host_listings_count	availability_365
##	0.00000000	0.00000000

Tenemos valores faltantes en las columnnas `name`, `host_name`, `last_review` y `reviews_per_month`, sin embargo, solo estas dos últimas estan por encima del umbral seguro (5%), lo que podría indicarnos a priori que son variables que deben eliminarse porque no aportarán al análisis. Sin embargo, esta es una

decisión que debe tomarse con un mayor análisis de estos registros.

Haremos uso de la función `md.pattern` del paquete `mice`, que nos brinda visualmente el patrón de los datos faltantes, para un mejor entendimiento de estos.

```
md.pattern(airbnb, plot = TRUE, rotate.names=TRUE)
```



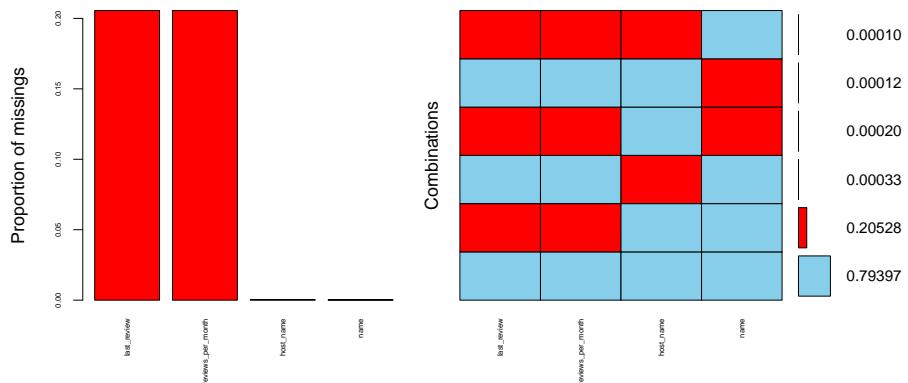
```
##      id host_id neighbourhood_group neighbourhood latitude longitude room_type
## 38821  1       1                   1           1       1       1       1       1
## 10037  1       1                   1           1       1       1       1       1
## 16     1       1                   1           1       1       1       1       1
## 5      1       1                   1           1       1       1       1       1
## 6      1       1                   1           1       1       1       1       1
## 10    1       1                   1           1       1       1       1       1
## 0      0       0                   0           0       0       0       0       0
##      price minimum_nights number_of_reviews calculated_host_listings_count
## 38821   1             1                 1                         1
## 10037   1             1                 1                         1
## 16     1             1                 1                         1
## 5      1             1                 1                         1
## 6      1             1                 1                         1
## 10    1             1                 1                         1
## 0      0             0                 0                         0
##      availability_365 name host_name last_review reviews_per_month
## 38821            1     1       1           1               1       0
```

```
## 10037      1   1     1      0      0   2
## 16        1   1     0      1      1   1
## 5         1   1     0      0      0   3
## 6         1   0     1      1      1   1
## 10       1   0     1      0      0   3
##          0  16    21  10052 10052 20141
```

El patrón nos indica que 38.821 registros no tienen datos faltantes, que los datos faltantes se encuentran en las columnas `name`, `host_name`, `last_review` y `reviews_per_month` (como habíamos encontrado anteriormente), con 16, 21, 10.052 y 10.052 registros, respectivamente. Adicionalmente, el mayor número de registros con datos faltantes (10.037) se encuentran en el patrón que solo contiene `na` en las columnas `last_review` y `reviews_per_month` y solo hay tres filas que contienen más de un valor perdido y de esas solo dos contienen más de dos valores perdidos.

Haciendo uso del paquete VIM, podemos ver la proporción de datos faltantes graficamente. Por cuestión de espacio y mejor visualización del gráfico, trabajaremos con el datafram solo con la columnas identificadas anteriormente que tienen datos faltantes

```
airbnb_columns=airbnb[,c("name","host_name","last_review","reviews_per_month")]
aggr(airbnb_columns, numbers=TRUE, sortVars=TRUE, labels=names(data), cex.axis=.5, gap=
```



```
##
##  Variables sorted by number of missings:
##          Variable      Count
##        last_review 0.2055833930
##  reviews_per_month 0.2055833930
##        host_name 0.0004294918
##          name 0.0003272318
```

El gráfico de barras anterior, nos muestra que las columnas `last_review` y `reviews_per_month` representan la mayor proporción de datos faltantes y la

proporción para la columnas `name` y `host_name` no es significativa. Este nuevo patrón nos complementa el anterior obtenido con el paquete `mice`puesto que nos indica adicionalmente que el 79,4% de los datos no tienen datos perdidos y nos muestra la proporción de filas que tienen un determinado patrón de datos perdidos, por ejemplo, el 20,52% tienen el patron de datos perdidos solo en las columnas en las columnas `last_review` y `reviews_per_month`.

El número de datos perdidos en el dataframe es bastante significativo (20.6%), sin embargo, al analizar lo que significan las columnas que los tienen, vemos por un lado que para el análisis posterior las columnas `name` y `host_name` no son necesarias y pueden eliminarse.

```
borrar = c("name", "host_name")
airbnb = airbnb[, !(names(airbnb) %in% borrar)]
head(airbnb)

##      id host_id neighbourhood_group neighbourhood latitude longitude
## 1 2539     2787             Brooklyn    Kensington 40.64749 -73.97237
## 2 2595     2845            Manhattan      Midtown 40.75362 -73.98377
## 3 3647     4632            Manhattan       Harlem 40.80902 -73.94190
## 4 3831     4869             Brooklyn   Clinton Hill 40.68514 -73.95976
## 5 5022     7192            Manhattan   East Harlem 40.79851 -73.94399
## 6 5099     7322            Manhattan Murray Hill 40.74767 -73.97500
##      room_type price minimum_nights number_of_reviews last_review
## 1 Private room   149           1                 9 2018-10-19
## 2 Entire home/apt  225           1                45 2019-05-21
## 3 Private room   150           3                 0 <NA>
## 4 Entire home/apt   89           1                270 2019-07-05
## 5 Entire home/apt   80           10                9 2018-11-19
## 6 Entire home/apt   200           3                74 2019-06-22
##      reviews_per_month calculated_host_listings_count availability_365
## 1              0.21                      6                  365
## 2              0.38                      2                  355
## 3                NA                      1                  365
## 4              4.64                      1                  194
## 5              0.10                      1                   0
## 6              0.59                      1                  129
```

Por otro lado, al analizar los registros faltantes en las columnas `last_review` y `reviews_per_month`, encontramos que todos corresponden a aquellos registros donde no existen una evaluación por lo tanto, el manejo de estos datos es simple y se procede de la siguiente manera, se asigna un 0 en la columna `reviews_per_month` y se elimina la columna `last_review` por no tener valor significativo para nuestro análisis posterior.

```
borrar = c("last_review")
airbnb = airbnb[, !(names(airbnb) %in% borrar)]
```

```
airbnb %> mutate_at(airbnb, c("reviews_per_month"), ~replace(., is.na(.), 0))
head(airbnb)

##      id host_id neighbourhood_group neighbourhood latitude longitude
## 1 2539    2787           Brooklyn     Kensington 40.64749 -73.97237
## 2 2595    2845          Manhattan      Midtown 40.75362 -73.98377
## 3 3647    4632          Manhattan       Harlem 40.80902 -73.94190
## 4 3831    4869           Brooklyn Clinton Hill 40.68514 -73.95976
## 5 5022    7192          Manhattan   East Harlem 40.79851 -73.94399
## 6 5099    7322          Manhattan Murray Hill 40.74767 -73.97500
##      room_type price minimum_nights number_of_reviews reviews_per_month
## 1 Private room   149            1                 9        0.21
## 2 Entire home/apt 225            1                45        0.38
## 3 Private room   150            3                 0        0.00
## 4 Entire home/apt  89            1               270        4.64
## 5 Entire home/apt  80            10                9        0.10
## 6 Entire home/apt 200            3                74        0.59
##      calculated_host_listings_count availability_365
## 1                           6             365
## 2                           2             355
## 3                           1             365
## 4                           1             194
## 5                           1              0
## 6                           1            129
```

En este punto nuestros datos ya no tienen valores faltantes y trabajaremos en adelante con un dataframe de 48.895 y 13 variables.

```
apply(airbnb, 2, pMiss)
```

```
##      id host_id
## 0          0
##      neighbourhood_group neighbourhood
## 0          0
##      latitude longitude
## 0            0
##      room_type price
## 0            0
##      minimum_nights number_of_reviews
## 0            0
##      reviews_per_month calculated_host_listings_count
## 0            0
##      availability_365
## 0            0
```

2.3 Creación en base de datos

El dataframe sin datos faltantes generado en la sección anterior debe cargarse en una base de datos en Heroku Postgress. Para esto primero debemos conectarnos a ella, usando la función dbConnect() con los datos apropiados.

```
con <- dbConnect(RPostgres::Postgres(),
                 dbname = "d41lsl8qgestjf",
                 host = "ec2-3-229-43-149.compute-1.amazonaws.com",
                 port = 5432,
                 user = "uqtxfaqjjcxggw",
                 password = "916d311356954de6a99118d13578bb9d1b47bdc86cb8360a60b9606293bd882d")
```

Una vez tengamos establecida la conexión, insertamos los datos en la tabla `airbnb`, usando la función `dbWriteTable()`

```
dbWriteTable(con, 'airbnb', airbnb, row.names=FALSE, overwrite=TRUE)
```

Verifiquemos que podamos leer los datos, através de la función `dbGetQuery()`

```
df = dbGetQuery(con, "SELECT * FROM airbnb")
summary(df)
```

```
##      id          host_id    neighbourhood_group neighbourhood
##  Min.   : 2539   Min.   : 2438   Length:48895           Length:48895
##  1st Qu.: 9471945  1st Qu.: 7822033  Class :character   Class :character
##  Median :19677284   Median : 30793816  Mode  :character   Mode  :character
##  Mean   :19017143   Mean   : 67620011
##  3rd Qu.:29152178   3rd Qu.:107434423
##  Max.   :36487245   Max.   :274321313
##      latitude       longitude     room_type        price
##  Min.   :40.50   Min.   :-74.24   Length:48895       Min.   : 0.0
##  1st Qu.:40.69   1st Qu.:-73.98   Class :character   1st Qu.: 69.0
##  Median :40.72   Median :-73.96   Mode  :character   Median : 106.0
##  Mean   :40.73   Mean   :-73.95
##  3rd Qu.:40.76   3rd Qu.:-73.94
##  Max.   :40.91   Max.   :-73.71       Mean   : 152.7
##                                         3rd Qu.: 175.0
##                                         Max.   :10000.0
##      minimum_nights   number_of_reviews reviews_per_month
##  Min.   : 1.00   Min.   : 0.00   Min.   : 0.000
##  1st Qu.: 1.00   1st Qu.: 1.00   1st Qu.: 0.040
##  Median : 3.00   Median : 5.00   Median : 0.370
##  Mean   : 7.03   Mean   :23.27   Mean   : 1.091
##  3rd Qu.: 5.00   3rd Qu.:24.00   3rd Qu.: 1.580
##  Max.   :1250.00  Max.   :629.00   Max.   :58.500
##      calculated_host_listings_count availability_365
##  Min.   : 1.000           Min.   : 0.0
##  1st Qu.: 1.000           1st Qu.: 0.0
##  Median : 1.000           Median : 45.0
```

```
##  Mean    : 7.144          Mean    :112.8
##  3rd Qu.: 2.000          3rd Qu.:227.0
##  Max.    :327.000        Max.    :365.0
```

Chapter 3

Análisis Exploratorio de los datos

En este capítulo abordaremos el análisis exploratorio de los dato *EDA* para el listado de Airbnb de la ciudad de Nueva York en el año 2019.

Resumen de los datos

Empezamos el análisis exploratorio de nuestros datos con las estadísticas de resumen, haciendo uso de la función `summary` para los datos contenidos en la tabla `airbnb` de nuestra base de datos en Heroku

```
con <- dbConnect(RPostgres::Postgres(),
  dbname = "d41lsl8qgestjf",
  host = "ec2-3-229-43-149.compute-1.amazonaws.com",
  port = 5432,
  user = "uqtxfaqjjcxggw",
  password = "916d311356954de6a99118d13578bb9d1b47bdc86cb8360a60b9606293bd882d")

df = dbGetQuery(con, "SELECT * FROM airbnb")
summary(df)

##      id          host_id    neighbourhood_group neighbourhood
##  Min.   : 2539   Min.   : 2438   Length:48895       Length:48895
##  1st Qu.: 9471945  1st Qu.: 7822033  Class :character   Class :character
##  Median :19677284  Median : 30793816 Mode  :character   Mode  :character
##  Mean   :19017143  Mean   : 67620011
##  3rd Qu.:29152178  3rd Qu.:107434423
##  Max.   :36487245  Max.   :274321313
##      latitude     longitude    room_type           price
```

```

## Min. :40.50   Min. :-74.24   Length:48895      Min. : 0.0
## 1st Qu.:40.69  1st Qu.:-73.98   Class :character  1st Qu.: 69.0
## Median :40.72  Median :-73.96   Mode  :character  Median :106.0
## Mean   :40.73  Mean  :-73.95   Mean   :152.7
## 3rd Qu.:40.76  3rd Qu.:-73.94   3rd Qu.: 175.0
## Max.  :40.91   Max.  :-73.71   Max.  :10000.0
## minimum_nights   number_of_reviews reviews_per_month
## Min. : 1.00   Min. : 0.00   Min. : 0.000
## 1st Qu.: 1.00   1st Qu.: 1.00   1st Qu.: 0.040
## Median : 3.00   Median : 5.00   Median : 0.370
## Mean   : 7.03   Mean  :23.27   Mean  : 1.091
## 3rd Qu.: 5.00   3rd Qu.:24.00   3rd Qu.: 1.580
## Max.  :1250.00  Max.  :629.00   Max.  :58.500
## calculated_host_listings_count availability_365
## Min. : 1.000   Min. : 0.0
## 1st Qu.: 1.000   1st Qu.: 0.0
## Median : 1.000   Median : 45.0
## Mean   : 7.144   Mean  :112.8
## 3rd Qu.: 2.000   3rd Qu.:227.0
## Max.  :327.000  Max.  :365.0

```

De las columnas de nuestro dataframe, podemos decir por ejemplo que el precio de una noche de los airbnb oscila entre 0 y 10.000 dólares con un promedio de 152.7 dólares, más adelante revisaremos si un precio diario de 10.000 dólares es o no un dato atípico. Adicionalmente, los alojamientos se pueden reservar desde 1 noche sin embargo hay algunos cuyas noches mínimas son de 1.250, alrededor de 3.5 años, esto también es un candidato a dato atípico que será revisado en la siguiente sección. Por otro lado, existen anfitriones que tienen hasta 327 alojamientos en la región.

A pesar, que las columnas `id`, `host_id`, `latitude` y `longitude` son numéricas, no son relevante las métricas de mínimo, máximo, media y cuartiles.

Evaluando los valores que pueden tomar las variables categóricas usando la función `unique`, encontramos que los tipos de alojamiento disponibles son: Habitaciones privadas (“Private room”), Apartamentos Completos (“Entire home/apt”) o Habitaciones compartidas (“Shared room”) y tenemos grupos de vecindarios como Brooklyn, Manhattan, Qeens, Staten Island y Bronx y en total 221 vecindarios disponibles para alojamiento.

```
unique(df$room_type)
```

```
## [1] "Private room"    "Entire home/apt" "Shared room"
```

```
unique(df$neighbourhood_group)
```

```
## [1] "Brooklyn"        "Manhattan"       "Queens"          "Staten Island"
## [5] "Bronx"
```

```
length(unique(df$neighbourhood))
```

```
## [1] 221
```

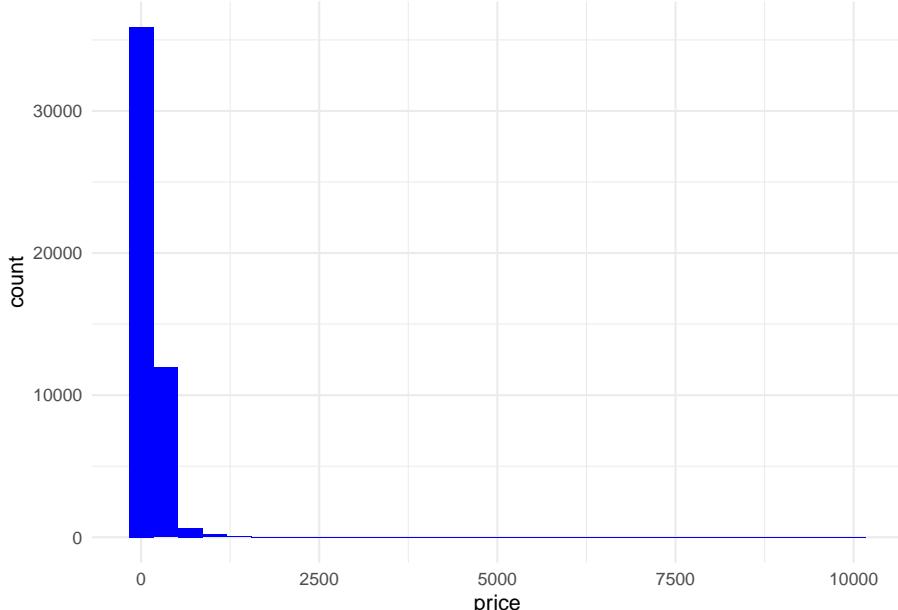
Datos Atípicos

Tenemos sospechas que existen datos atípicos en las columnas `price` y `'minimum_nights'`. Al analizar los histogramas y boxplots encontramos que estos abarcan la mayor parte del rango de las variables.

```
par(mfrow = c(1, 2))
```

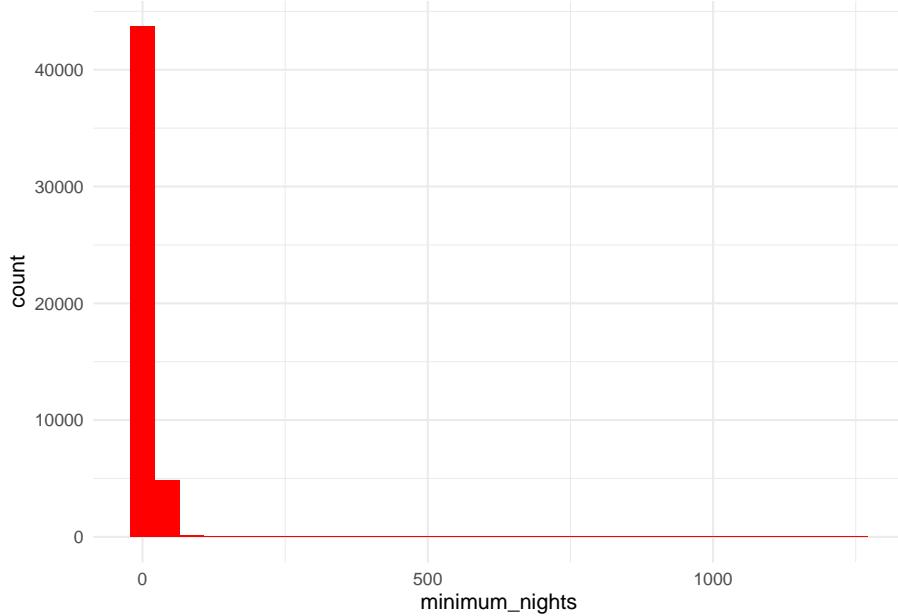
```
ggplot(df) +
  aes(x = price) +
  geom_histogram(fill = "blue") +
  theme_minimal()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

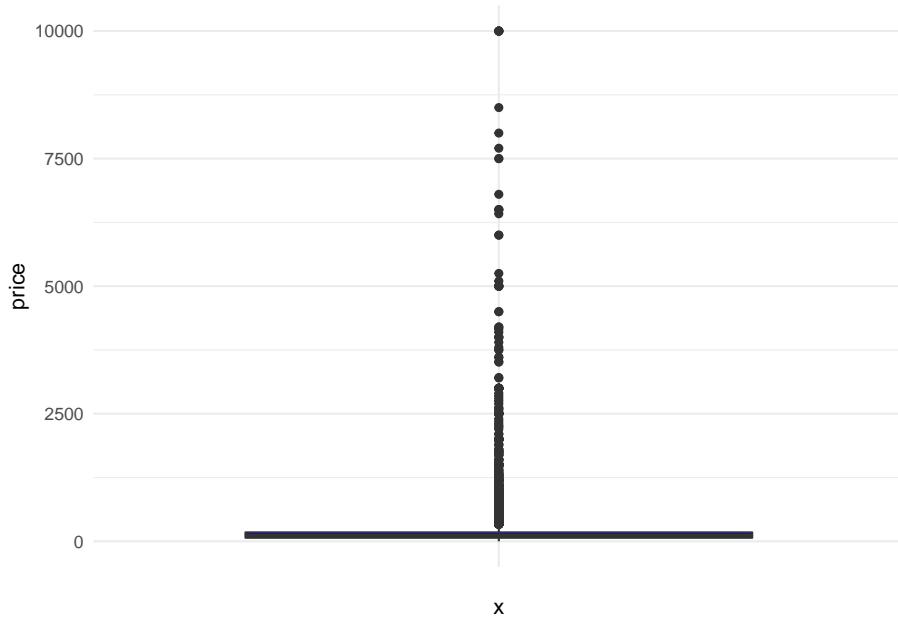


```
ggplot(df) +
  aes(x = minimum_nights) +
  geom_histogram(fill = "red") +
  theme_minimal()
```

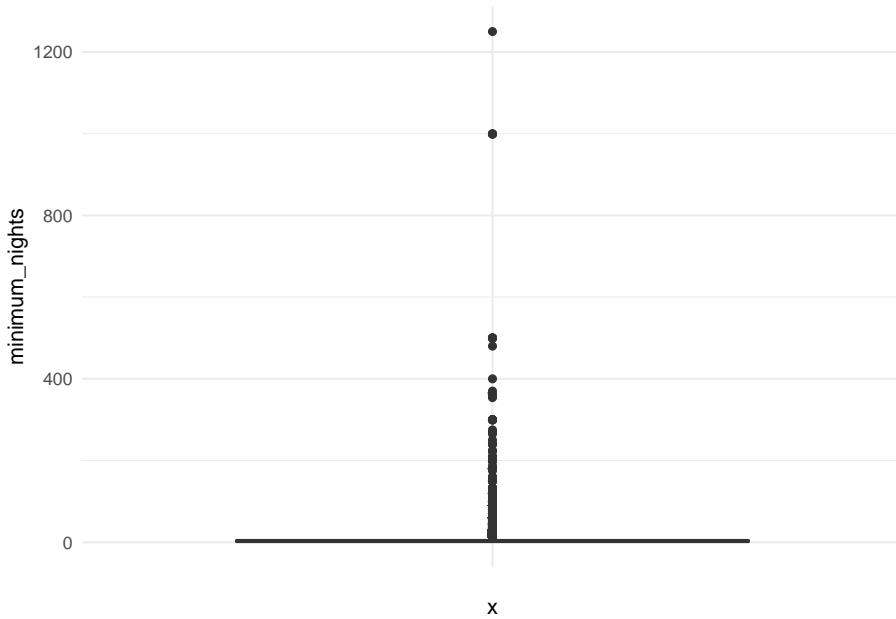
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
ggplot(df) +  
  aes(x = "", y = price) +  
  geom_boxplot(fill = "blue") +  
  theme_minimal()
```



```
ggplot(df) +
  aes(x = "", y = minimum_nights) +
  geom_boxplot(fill = "red") +
  theme_minimal()
```



Según la función `boxplot.stats()$out` que se basa en el criterio IQR, los valores atípicos para el precio son aquellos mayores que 335 mientras que para el mínimo de noches el umbral está en 12.

```
min(boxplot.stats(df$price)$out)
```

```
## [1] 335
min(boxplot.stats(df$minimum_nights)$out)
```

```
## [1] 12
```

Si revisamos por el criterio de los percentiles, teniendo en cuenta que las observaciones por fuera del intervalo formado por los percentiles 2.5 y 97.5 se consideran posibles valores atípicos, encontramos que para el precio son potenciales datos atípicos aquellos por debajo de 30 y por encima de 799 y en el caso de las noches mínimas, aquellas observaciones por debajo de 1 y por encima de 30.

```
c(quantile(df$price, 0.025), quantile(df$price, 0.975))
```

```
## 2.5% 97.5%
```

```
##      35    500
c(quantile(df$minimum_nights, 0.025), quantile(df$minimum_nights, 0.975))

##  2.5% 97.5%
##      1     30
```

Otro método, es el filtro de Hampel que determina que es un dato atípico si el precio está por encima de 244 y las noches minimas superan las 9.

```
c(median(df$price) - 3 * mad(df$price, constant = 1), median(df$price) + 3 * mad(df$price))

## [1] -32 244
c(median(df$minimum_nights) - 3 * mad(df$minimum_nights, constant = 1), median(df$minimum_nights))

## [1] -3 9
```

Siendo conservadores, eliminaremos la menor cantidad de datos atípicos posibles, que se consiguen al usar el umbral dado por el criterio de los percentiles. Siendo así, estamos eliminando el 2.13% de nuestros datos, resultando un dataframe con 1.044 registros menos.

```
nrow(df)

## [1] 48895
nrow(df[df$price>335, ])/nrow(df)*100.00

## [1] 6.06197
nrow(df[df$price>500, ])/nrow(df)*100.00

## [1] 2.135188
nrow(df[df$price>244, ])/nrow(df)*100.00

## [1] 13.26311
```

Una vez eliminados estos datos atípicos, volveos a extraer las estadísticas de resumen

```
df_out = df[df$price<500, ]
summary(df_out)

##           id          host_id      neighbourhood_group neighbourhood
## Min.   : 2539   Min.   : 2438   Length:47660          Length:47660
## 1st Qu.: 9463850  1st Qu.: 7778878  Class :character    Class :character
## Median :19623162  Median : 30567770 Mode  :character    Mode  :character
## Mean   :18979918  Mean   : 67134263
## 3rd Qu.:29058184  3rd Qu.:107216950
## Max.   :36487245  Max.   :274321313
##           latitude       longitude      room_type            price
##
```

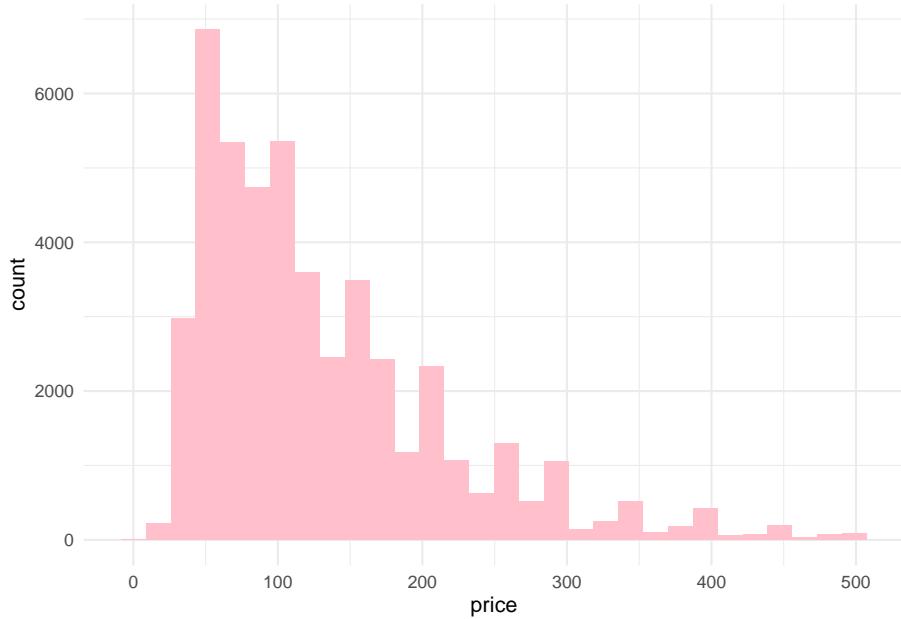
```

## Min.   :40.50   Min.   :-74.24   Length:47660      Min.   :  0.0
## 1st Qu.:40.69   1st Qu.:-73.98   Class :character   1st Qu.: 68.0
## Median :40.72   Median :-73.96   Mode  :character   Median :100.0
## Mean    :40.73   Mean   :-73.95          Mean    :130.1
## 3rd Qu.:40.76   3rd Qu.:-73.94          3rd Qu.:170.0
## Max.    :40.91   Max.   :-73.71          Max.    :499.0
## minimum_nights     number_of_reviews reviews_per_month
## Min.   : 1.000   Min.   : 0.00   Min.   : 0.000
## 1st Qu.: 1.000   1st Qu.: 1.00   1st Qu.: 0.040
## Median : 2.000   Median : 5.00   Median : 0.380
## Mean   : 6.978   Mean   :23.59   Mean   : 1.102
## 3rd Qu.: 5.000   3rd Qu.:24.00   3rd Qu.: 1.610
## Max.   :1250.000  Max.   :629.00   Max.   :58.500
## calculated_host_listings_count availability_365
## Min.   : 1.000           Min.   :  0
## 1st Qu.: 1.000           1st Qu.:  0
## Median : 1.000           Median : 42
## Mean   : 7.096           Mean   :111
## 3rd Qu.: 2.000           3rd Qu.:221
## Max.   :327.000           Max.   :365

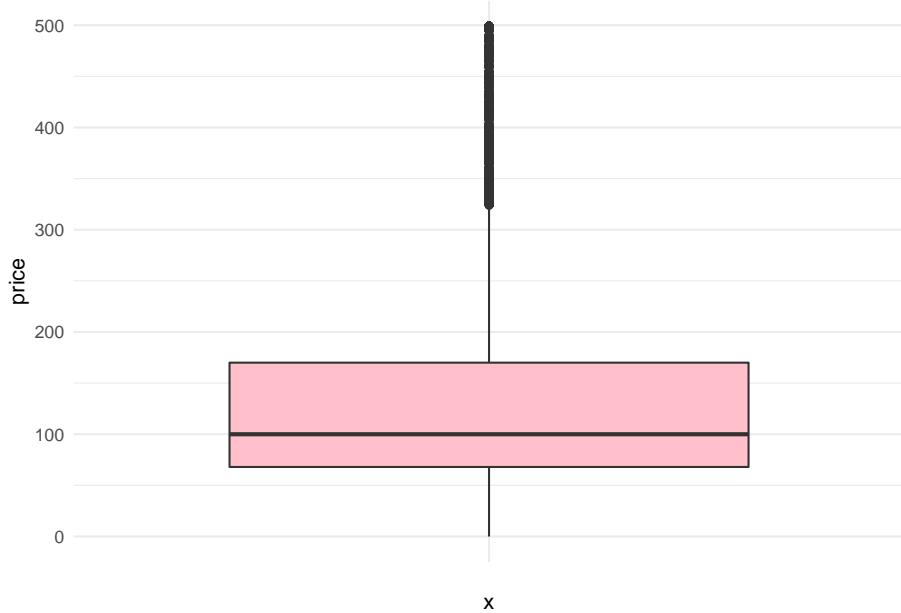
par(mfrow = c(1, 2))

ggplot(df_out) +
  aes(x = price) +
  geom_histogram(fill = "pink") +
  theme_minimal()

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
ggplot(df_out) +  
  aes(x = "", y = price) +  
  geom_boxplot(fill = "pink") +  
  theme_minimal()
```

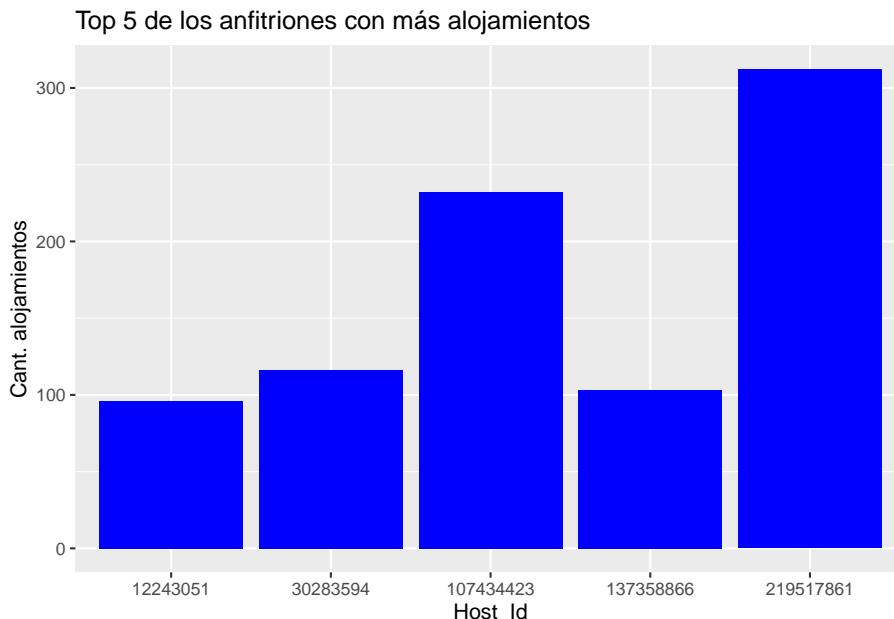


Análisis del anfitrion

A través de la función de agrupación `group_by`, podemos obtener los anfitriones con más alojamientos disponibles, encontrando que el mayor tiene 272 lo que no es consistente con el máximo valor encontrado en la columna `calculated_host_listings_count` debido a la eliminación de datos atípicos realizados anteriormente, es por esto, que esta columna, se eliminará del dataframe.

```
df_out %>%
  group_by(host_id) %>%
  summarise(n = n()) %>%
  arrange(desc(n)) %>%
  head(5) -> top_hostid

## `summarise()` ungrouping output (override with `.`groups` argument)
ggplot(data= top_hostid, aes(x = factor(host_id), y=n)) +
  geom_bar(stat="identity", fill = "blue") +
  xlab("Host_Id") + ylab("Cant. alojamientos") +
  ggtitle("Top 5 de los anfitriones con más alojamientos")
```



```
borrar = c("calculated_host_listings_count")
df_out = df_out[, !(names(df_out) %in% borrar)]
```

Ahora, revisemos la distribución de alojamientos por grupos de vecindarios.

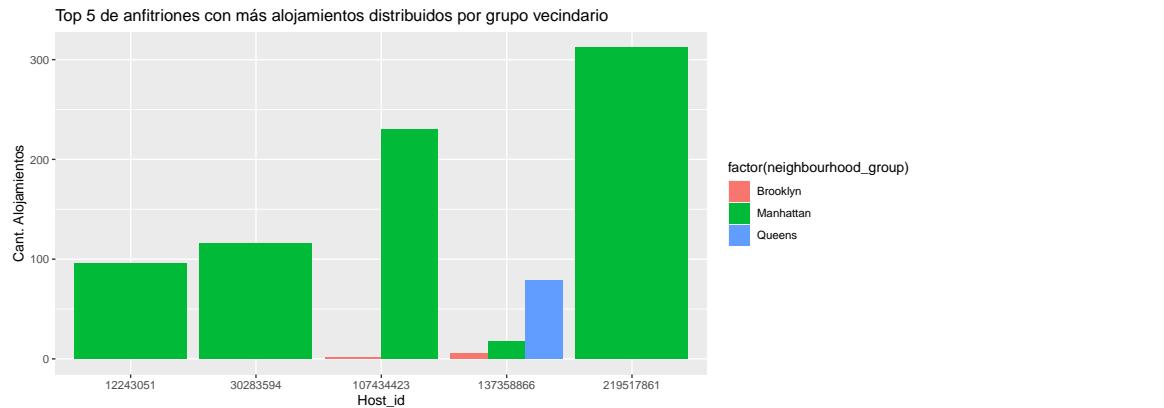
```

df_host = filter(df_out, host_id %in% top_hostid$host_id)

df_host%>%
  group_by(host_id, neighbourhood_group) %>%
  summarise(n = n()) %>%
  ggplot(aes(x=factor(host_id), y=n, fill=factor(neighbourhood_group))) +
  geom_bar(stat="identity", position="dodge")+
  xlab("Host_id") + ylab("Cant. Alojamientos") +
  ggtitle("Top 5 de anfitriones con más alojamientos distribuidos por grupo vecindario")
  theme()

## `summarise()` regrouping output by 'host_id' (override with `.`groups` argument)

```



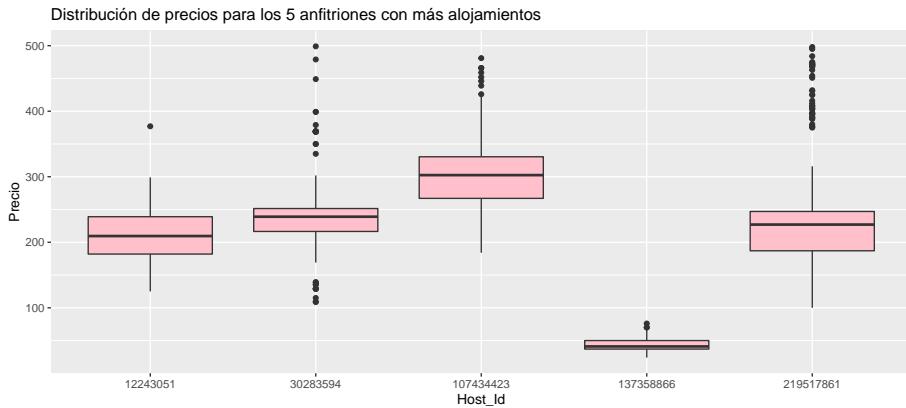
A pesar, que los grupos de vecindarios disponibles corresponden a Brooklyn, Manhattan, Qeens, Staten Island y Bronx, vemos que los anfitriones con más alojamientos no tienen disponibilidad en Staten Island y Bronx y exceptuando el host 137358866, la mayoría de estos alojamientos se encuentran en Manhattan y solo dos de estos cinco tienenalojamientos en Brooklyn.

Si revisamos la distribución de los precios de estos cinco anfitriones, atraves, de un boxplot, tenemos que el anfitrion 107434423 tiene el mayor precio promedio y el anfitrion 137358866 tiene todos sus alojamnientos en precios similares y más bajos en comparación con los otros.

```

df_host%>%
  ggplot(aes(x=factor(host_id), y=price)) +
  geom_boxplot(fill = "pink")+
  xlab("Host_Id") + ylab("Precio") +
  ggtitle("Distribución de precios para los 5 anfitriones con más alojamientos")

```



```
theme()
```

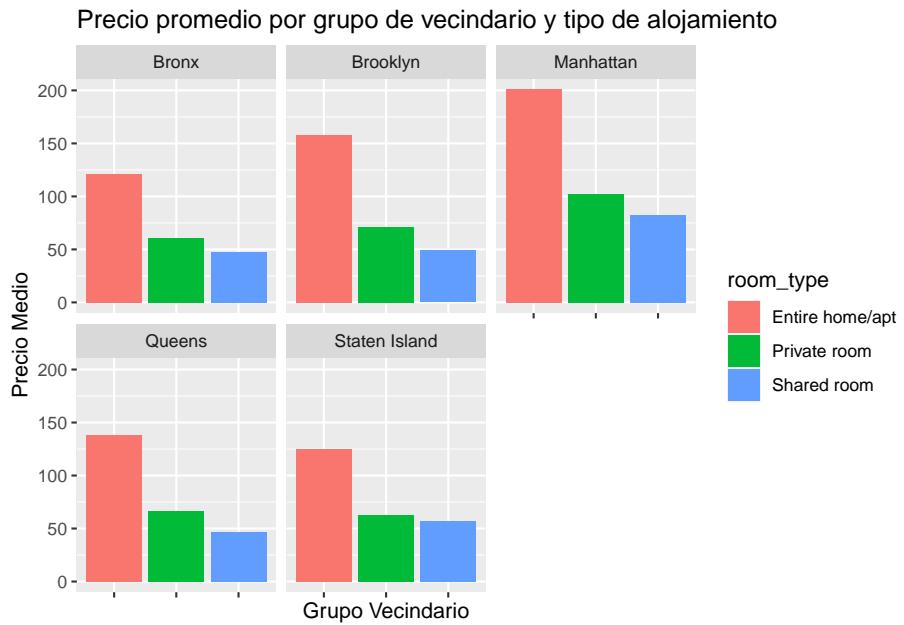
```
## Named list()
## - attr(*, "class")= chr [1:2] "theme" "gg"
## - attr(*, "complete")= logi FALSE
## - attr(*, "validate")= logi TRUE
```

Análisis de los precios

Si ahora revisamos los precios, vemos que el mayor precio promedio por noche está en la zona de Manhattan para los tres tipos de alojamiento, adicionalmente y como era de esperarse, es más costoso un alojamiento completo, seguido de una habitación privada y por último una habitación compartida, aunque no hay una gran diferencia entre los valores promedios de las habitaciones. Este patrón se mantiene igual, independiente del grupo de vecindario al que pertenezca.

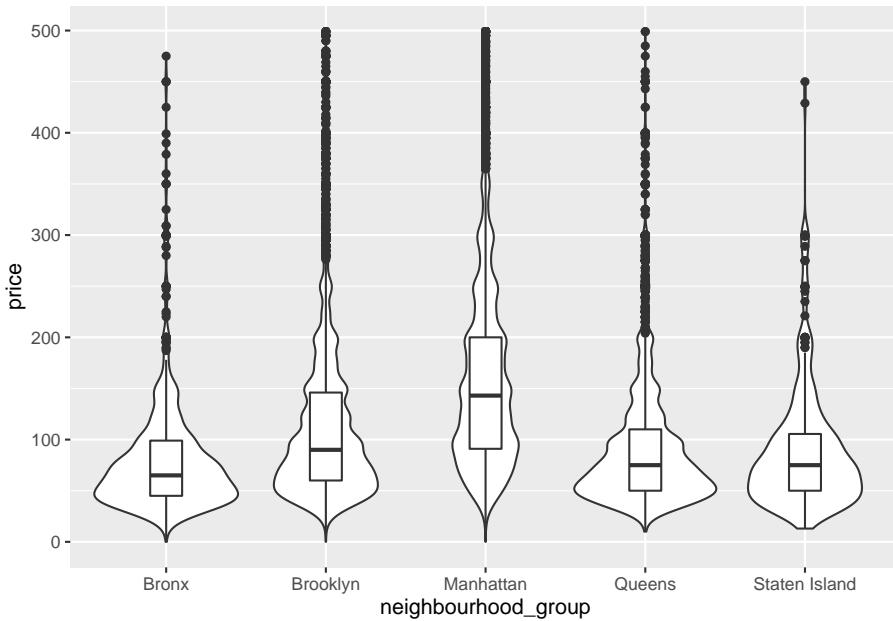
```
df_out %>%
  group_by(neighbourhood_group, room_type)%>%
  summarise(m = mean(price)) -> group_type

## `summarise()` regrouping output by 'neighbourhood_group' (override with `groups` argument)
group_type%>%
  ggplot(aes(x=room_type, y=m, fill=room_type)) +
  geom_bar(stat="identity")+
  facet_wrap(~neighbourhood_group)+
  xlab("Grupo Vecindario") + ylab("Precio Medio") +
  ggtitle("Precio promedio por grupo de vecindario y tipo de alojamiento")+
  theme(axis.text.x=element_blank())
```



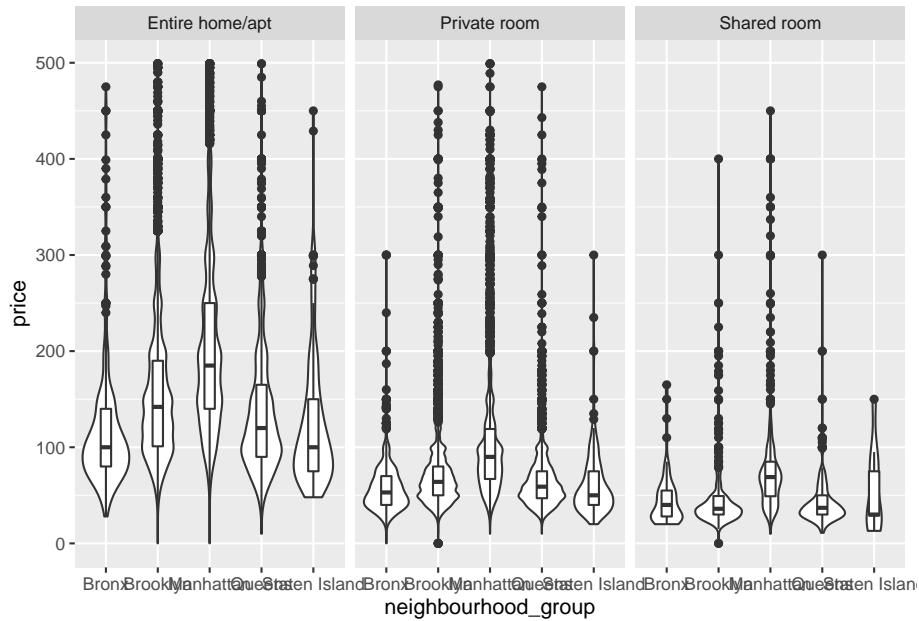
El grupo de vecindario con el precio promedio más alto es Manhattan y en este grupo se manejan el mayor rango de precios. Los precios para Queen y Staten Island tienen distribuciones similares.

```
ggplot(data=df_out, mapping = aes(x=neighbourhood_group, y=price)) +
  geom_violin()+
  geom_boxplot(width=0.2)
```



Si agrupamos por tipo de alojamiento, vemos que los valores por habitación compartida en Staten Island presentan una distribución distintas a las demás con un rango mayor y un sesgo hacia la izquierda. En cuanto a las habitaciones privadas tenemos una alta concentración de valores atípicos. En los alojamientos completos por grupo el precio es consistente, sin muchos valores atípicos.

```
ggplot(data=df_out, mapping = aes(x=neighbourhood_group, y=price)) +
  geom_violin()+
  geom_boxplot(width=0.2)+
  facet_wrap(~room_type)
```



Visualización en mapas

Para los datos de interés, contamos con la ubicación (latitud, longitud) de los alojamientos en Nueva York. Revisaremos la distribución de estos en un mapa, a través de la función `gg_map`, ubicándonos precisamente en Nueva York, encontrando que la menor cantidad de alojamientos se encuentra en “Staten Island”

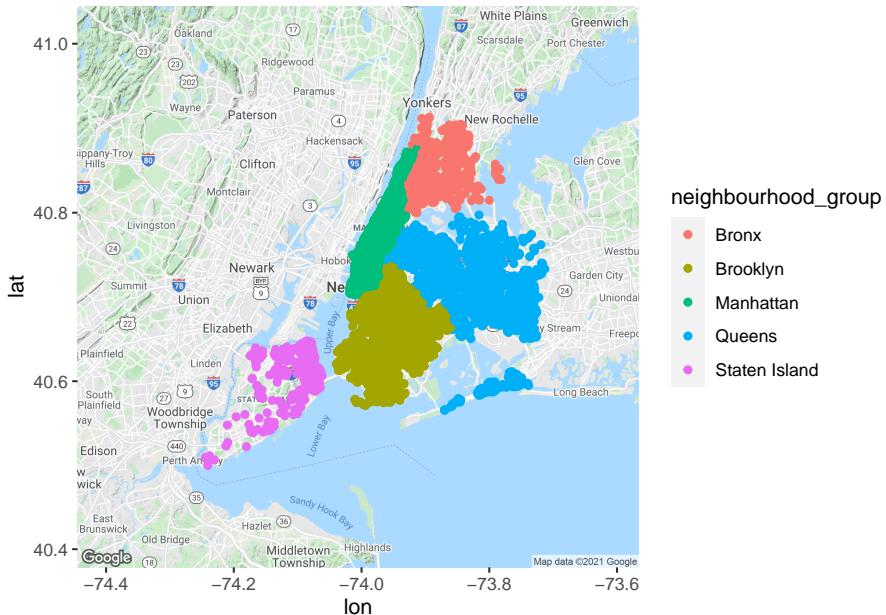
```
mykey = "AIzaSyCRgNUY6U40KR4MHy0RHkSUsSRLkE_0i0"
register_google(key = mykey)

myLocation <- "Nueva York"

myMap <- get_map(location = myLocation, zoom = 10)
```

```
## Source : https://maps.googleapis.com/maps/api/staticmap?center=Nueva%20York&zoom=10
```

```
## Source : https://maps.googleapis.com/maps/api/geocode/json?address=Nueva+York&key=xxxxxx
ggmap(myMap) + geom_point(data=df_out, aes(x = longitude, y = latitude, colour= neighbourhood_group))
```



En Bronx, hay pocos precios altos en los alojamientos y los que existen se encuebntran a los alrededores de la localidad. Si nos enfocamos en Manhattan, vemos que hay una clara división en los precios, los menores se encuentran al norte y los mayores al sur.

En general hay pocos alojamientos con precios altos en comparación con los de precios bajos.

```
ggmap(myMap) +
  geom_point(data=df_out, aes(x = longitude, y = latitude, colour= price))+
  scale_color_gradientn(colours = rainbow(5))+
```

facet_wrap(~neighbourhood_group)

