

# Ciencia de datos geograficos

Silvia Laceiras

Felipe Sodré M. Barros

Fabián Rechberger

Vamos de viaje!

Vamos de Viaje!



# pero...

- Cuanto nos va a costar?
- Donde nos vamos a quedar?
- Qué tipo de alquiler podremos buscar?

Vamos a analizar los datos de

# instalando algunos paquetes necesarios

Vamos a ocupar el paquete `geobr` para acceder al dato espacial del municipio de Rio de Janeiro. Y vamos a ocupar el paquete `tidyverse` para la limpieza de datos:

```
install.packages("geobr")  
install.packages("tidyverse")  
install.packages("ggplot2")
```

# Cargando los paquetes que vamos a ocupar

```
library(sf)
```

```
## Linking to GEOS 3.10.2, GDAL 3.4.1, PROJ 8.2.1; sf_use_s2() is TRUE
```

```
library(tmap)
```

```
library(geobr)
```

```
library(tidyverse)
```

```
## — Attaching core tidyverse packages ————— tidyverse 2.0.0
```

```
## ✓ dplyr      1.1.2      ✓ readr      2.1.4
```

```
## ✓ forcats   1.0.0      ✓ stringr    1.5.0
```

```
## ✓ ggplot2    3.4.2      ✓ tibble     3.2.1
```

```
## ✓ lubridate  1.9.2      ✓ tidyr      1.3.0
```

```
## ✓ purrr      1.0.1
```

```
## — Conflicts ————— tidyverse_conflicts()
```

```
## ✖ dplyr::filter() masks stats::filter()
```

```
## ✖ dplyr::lag()     masks stats::lag()
```

```
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all con
```

# Introducción

Pero más allá de prepararnos para el viaje, vamos a aprender:

- Cargar un dato CSV ;
- Algunas herramientas de limpieza de datos;
- Creación de gráficos (histograma y boxplot);



Qué es el proceso de limpieza de  
datos?



Qué es un archivo CSV?



# Cargando los datos de Airbnb

Vamos a cargar el dato csv con la función `read_csv`, del paquete `readr`, que nos brinda `tidyverse`.

```
datos <- read_csv("./Datos/AirbnbRJRentals_modificado.csv")
```

```
## Rows: 27507 Columns: 16
```

```
## — Column specification
```

---

```
## Delimiter: ","
```

```
## chr   (4): name, host_name, neighbourhood, room_type
```

```
## dbl  (11): id, host_id, latitude, longitude, price, minimum_nights, number_o.
```

```
## date  (1): last_review
```

```
##
```

```
## i Use `spec()` to retrieve the full column specification for this data.
```

```
## i Specify the column types or set `show_col_types = FALSE` to quiet this messag
```

# Datos del Airbnb

Lo interesante es que el dato de alquileres de Airbnb, brinda dos columns que nos puede ser interesante:

- latitude
- longitude

# convirtiendo el dataframe en un dato geográfico

Para convertir el `csv` en un dato geográfico tendremos que usar la función `st_as_sf`, informando cuales columnas tienen la información de coordenadas:

```
# transformando en geografico
```

```
(airbnb <- st_as_sf(datos, coords = c("longitude", "latitude")))
```

```
## Simple feature collection with 27507 features and 14 fields
```

```
## Geometry type: POINT
```


```
## Dimension: XY
```

```
## Bounding box: xmin: -43.70591 ymin: -23.07284 xmax: -43.1044 ymax: -22.74969
```

```
## CRS: NA
```

```
## # A tibble: 27,507 × 15
```

```
##      id name      host_id host_name neighbourhood room_type price minimum_nig
##      <dbl> <chr>      <dbl> <chr>      <chr>          <chr>      <dbl>      <d
##  1 1.05e 7 Sobra... 5.39e7 Quiá      Laranjeiras Entire h... 581
##  2 5.39e 7 Quadr... 3.34e8 Lucas      Copacabana Entire h... 898
##  3 7.83e17 Leme,... 4.92e8 Felipe     Leme        Entire h... 720
##  4 7.04e17 Suíte... 2.04e7 Júlio Ce... Botafogo    Private ... 599
##  5 7.83e17 Apart... 3.02e8 Lafe      Laranjeiras Entire h... 740
```



Una vez que ya tenemos los datos  
geográficos en formato *simple  
feature*, podríamos hacer un  
"mapita", no?



```
tm_shape(airbnb) +  
  tm_dots()
```

```
## Warning: Current projection of shape airbnb unknown. Long-lat (WGS84) is  
## assumed.
```



# Descargando datos del municipio de Rio de Janeiro

Para eso vamos a ocupar la función `get_municipality` del paquete `geobr`. Y aprovechamos y hacemos un "mapita", no?

```
(rj <- read_municipality(code_mun='RJ'))
```

```
## Using year 2010
```

```
##
```

```
Downloading: 770 B
```

```
Downloading: 770 B
```

```
Downloading: 1.6 kB
```

```
Downloading: 1.6 kB
```

```
Downloading: 1.8 kB
```

```
Downloading: 1.8 kB
```

```
Downloading: 1.8 kB
```

```
Downloading: 1.8 kB
```

```
Downloading: 1.8 kB
```

```
Downloading: 1.8 kB
```

```
Downloading: 1.9 kB
```

# Limpieza de datos: filtrado

Lo que descargamos son los municipios de la provincia de Rio. Pero queremos trabajar solamente con el município de Rio de Janeiro. Por eso, vamos a filtrar, de todos los municipios, aquellos que poseen el campo `name_muni` igual a 'Rio De Janeiro'

Vamos a ocupar la función `filter`:

```
rj <- filter(rj, name_muni == 'Rio De Janeiro')
```

ATENCIÓN: El símbolo `=` es de atribución. Lo que estamos haciendo es una consulta, por eso se ocupa el simbolo `==`;

A partir del mapa, ya se puede  
identificar un patrón en la  
distribución de los alquileres?

Empecemos el viaje: crear un mapa  
dinámico (webmap) con el  
municipio de Rio y los puntos de  
alquiler....

# Análisis exploratorio

# Análisis exploratorio

Una de las primeras cosas que se hace al acceder a algun conjunto de datos es entender qué columnas existen, que valores poseen,

# análisis exploratorio

Qué datos existen en la planilla (data frame)?

```
colnames(airbnb)
```

```
## [1] "id" "name"
## [3] "host_id" "host_name"
## [5] "neighbourhood" "room_type"
## [7] "price" "minimum_nights"
## [9] "number_of_reviews" "last_review"
## [11] "reviews_per_month" "calculated_host_listings_count"
## [13] "availability_365" "number_of_reviews_ltm"
## [15] "geometry"
```



# Resumo estatístico

```
summary(airbnb)
```

```
##           id           name           host_id           host_name
##  Min.      :1.788e+04   Length:27507   Min.      :    3607   Length:27507
##  1st Qu.:1.581e+07   Class :character   1st Qu.: 18151000   Class :character
##  Median :4.439e+07   Mode  :character   Median : 79141096   Mode  :character
##  Mean    :2.234e+17
##  3rd Qu.:6.351e+17
##  Max.    :7.916e+17
##
##  neighbourhood   room_type           price           minimum_nights
##  Length:27507     Length:27507   Min.      :    0.0   Min.      :    1.000
##  Class :character   Class :character   1st Qu.: 307.0   1st Qu.:    1.000
##  Mode  :character   Mode  :character   Median : 589.0   Median :    2.000
##
##  Mean    : 814.3   Mean    :    4.822
##  3rd Qu.:1000.0   3rd Qu.:    4.000
##  Max.    :4983.0   Max.    :   1125.000
##
##  number_of_reviews  last_review           reviews_per_month
##  Min.      :    0.0   Min.      :2012-02-21   Min.      : 0.010
##  1st Qu.:    0.0   1st Qu.:2022-08-14   1st Qu.: 0.190
```

# Qué información tenemos en *room\_type*?

```
unique(airbnb$room_type)
```

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

graficos con *ggplot2*

# Introducción al *ggplot2*

El `ggplot` tiene una estructura similar al `tmap`.

El dato de entrada debe figurar en la función `ggplot()` :

```
ggplot(airbnb)
```

# Introducción al *ggplot2*

Al dato de entrada se agregan las definiciones de visualización:

- `aes()` +
- `geom_()` +
- `theme_()`

```
ggplot(airbnb) + geom_histogram(aes(x=price))
```

# Introducción al *ggplot2*

## The Grammar of Graphics (1999)

El paquete `ggplot2`, tal cual `tmap` está basado en el abordaje de gramática de los gráficos.

Fonte: [texto en inglés](#)

# Introducción al *ggplot2*

## **ggplot2**

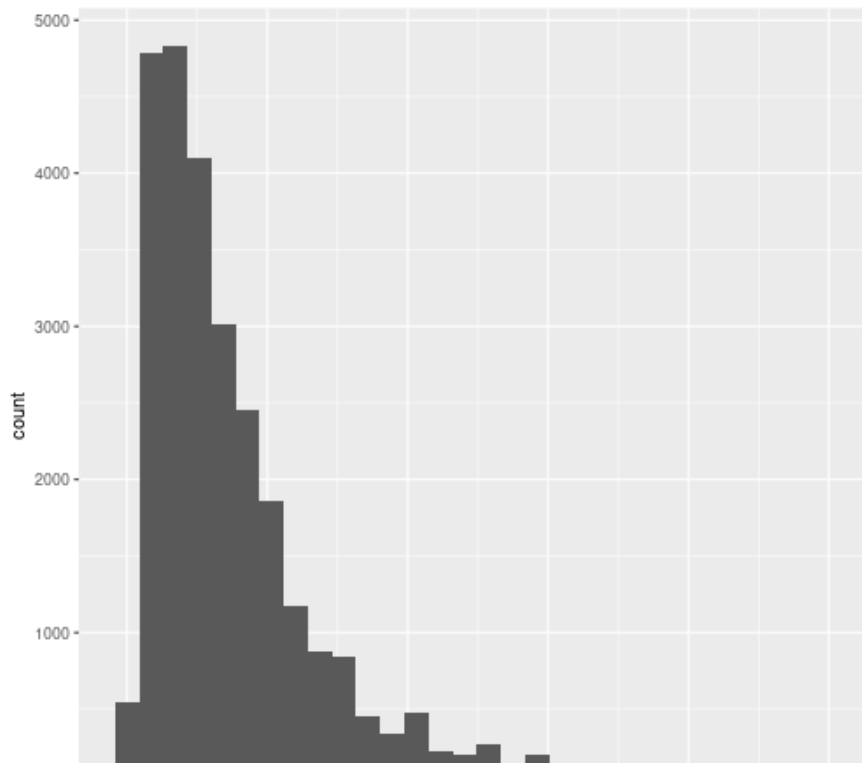
Fonte: <https://medium.com/tdebeus/think-about-the-grammar-of-graphics-when-improving-your-graphs-18e3744d8d18>

# Introducción al *ggplot2*



# Histograma ggplot2

El histograma es un gráfico en el cual se presenta la distribución de valores numéricos de una variable. Dicha variable tendrá sus valores numéricos representados en el eje horizontal ( y ) y el en eje vertical ( x ) la frecuencia de dicho valor.

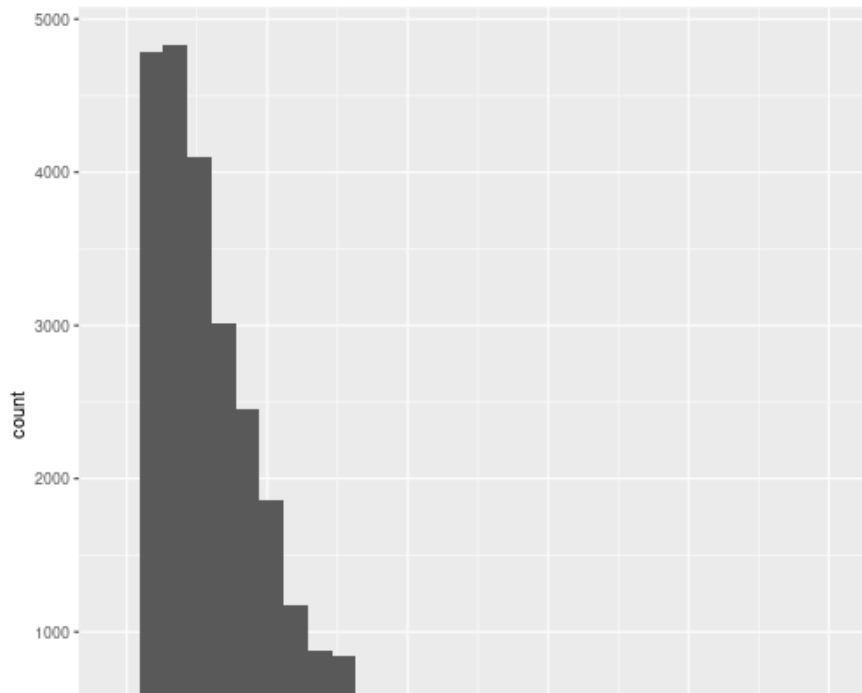


Volviendo a los datos de Rio de  
Janeiro...

# Como se distribuyen los datos de precio?

```
ggplot(airbnb) + geom_histogram(aes(x=price))
```

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



# Cual es el valor precio medio de los alquileres?

Para saber el valor medio, podemos usar la función `mean` y pasar a esta función el objeto que contiene los datos y cual columna queremos tener el valor medio calculado.

Para informar qué columna se debe usar, basta usar `$`:

```
objeto$columna
```

```
mean(airbnb$price)
```

```
## [1] 814.3001
```

Conociendo el *pipe* (%>%)

%>%

# %>%

El `pipe`, o tubería, es una manera de desencadenar una secuencia de acciones (funciones, transformaciones, etc);

Ejemplo, supongamos que queremos hacer un mapa de los alquileres que tengan el `room_type` igual a "Shared room". Podríamos hacer así:

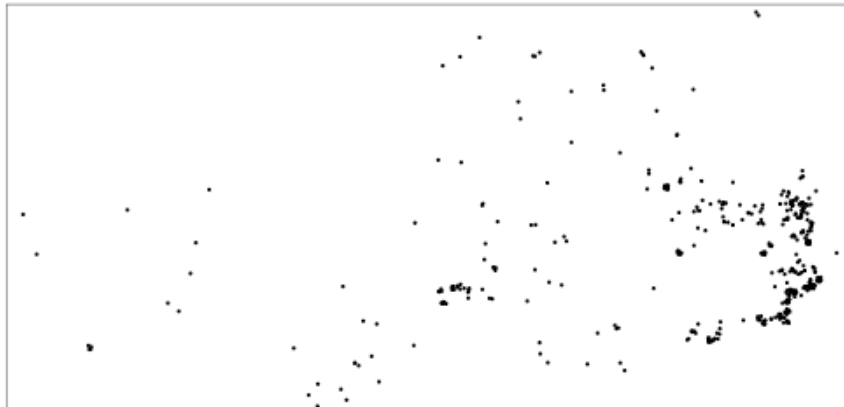
```
# primer paso
shared_rooms <- filter(airbnb,
                        room_type == 'Shared room')

# segundo paso
tm_shape(shared_rooms)+
  tm_dots()
```



O así:

```
airbnb %>%  
  filter(room_type == 'Shared room') %>%  
  tm_shape()+  
  tm_dots()
```





Ctrl + Shift + M: %>%

# Precio medio por barrio

Cómo podríamos calcular el valor medio de los alquileres por barrio?

Vamos a ocupar las funciones:

- `group_br()` (agrupar por) y
- `summarise()` (resumen);
- Y claro, el `%>%`

# Precio médio por bairro

Voy a agrupar los datos de `airbnb` por la columna `neighbourhood` (barrio) y aplicar el `summarise()`, calculando el valor medio de la columna `price`;

```
# Cual es el valor medio por barrio?
```

```
airbnb %>%  
  group_by(neighbourhood) %>%  
  summarise(  
    media = mean(price)  
  )
```

```
## Simple feature collection with 152 features and 2 fields
```

```
## Geometry type: GEOMETRY
```

```
## Dimension:      XY
```

```
## Bounding box:  xmin: -43.70591 ymin: -23.07284 xmax: -43.1044 ymax: -22.74969
```

```
## CRS:            NA
```

```
## # A tibble: 152 × 3
```

```
##   neighbourhood      media                                geome  
##   <chr>              <dbl>                                <GEOMET  
## 1 Abolição          466  MULTIPPOINT ((-43.29834 -22.88491), (-43.29698 -22  
## 2 Acari             250                                POINT (-43.33356 -22.814  
## 3 Alto da Boa Vista 1096. MULTIPPOINT ((-43.29683 -22.96931), (-43.29487 -22
```

# Precio médio por barrio

Para mejorar un poco la presentación, voy a usar al final la función `arrange()` y `desc()` para ordenar los valores de forma decreciente, así tendremos los valores más elevados primero:

```
airbnb %>% group_by(neighbourhood) %>% summarise(media = mean(price)) %>% ar
```

```
## Simple feature collection with 152 features and 2 fields
```

```
## Geometry type: GEOMETRY
```

```
## Dimension: XY
```

```
## Bounding box: xmin: -43.70591 ymin: -23.07284 xmax: -43.1044 ymax: -22.74969
```

```
## CRS: NA
```

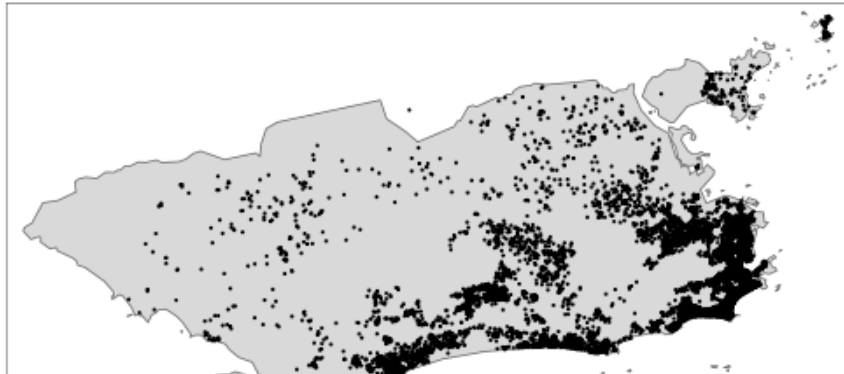
```
## # A tibble: 152 × 3
```

##	neighbourhood	media	geome
##	<chr>	<dbl>	<GEOMET
##	1 Anchieta	2776.	MULTIPOINT ((-43.40549 -22.82838), (-43.38878
##	2 Joá	1764.	MULTIPOINT ((-43.29633 -23.01625), (-43.29617
##	3 Ricardo de Albuquerque	1341.	MULTIPOINT ((-43.40667 -22.84112), (-43.40437
##	4 Lagoa	1230.	MULTIPOINT ((-43.22564 -22.97475), (-43.22555
##	5 Leblon	1218.	MULTIPOINT ((-43.23537 -22.98579), (-43.23481
##	6 Ipanema	1212.	MULTIPOINT ((-43.21555 -22.98483), (-43.21546
##	7 São Conrado	1176.	MULTIPOINT ((-43.27564 -23.00215), (-43.27528

Dónde podríamos quedarnos  
pagando el valor promedio?

# Dónde podríamos quedarnos con el valor promedio

```
airbnb_814 <- airbnb %>% filter(price <= 814)
tm_shape(rj)+
  tm_polygons() +
  tm_shape(airbnb_814) +
  tm_dots()
```



# Y si filtramos por algunos barrios?

Dos barrios bastante conocidos en Rio son Copacabana y Barra da Tijuca. Están cerca a la playa, pero representan ambientes distintos:

# Y si filtramos por algunos barrios?

Como tenemos la info del barrio, podríamos investigar los barrios que más nos interesa, usando el `filter()` y el operador `%in%`.

```
airbnb_814 <-  
  airbnb_814 %>%  
  filter(neighbourhood %in%  
         c("Copacabana", "Barra da Tijuca"))
```



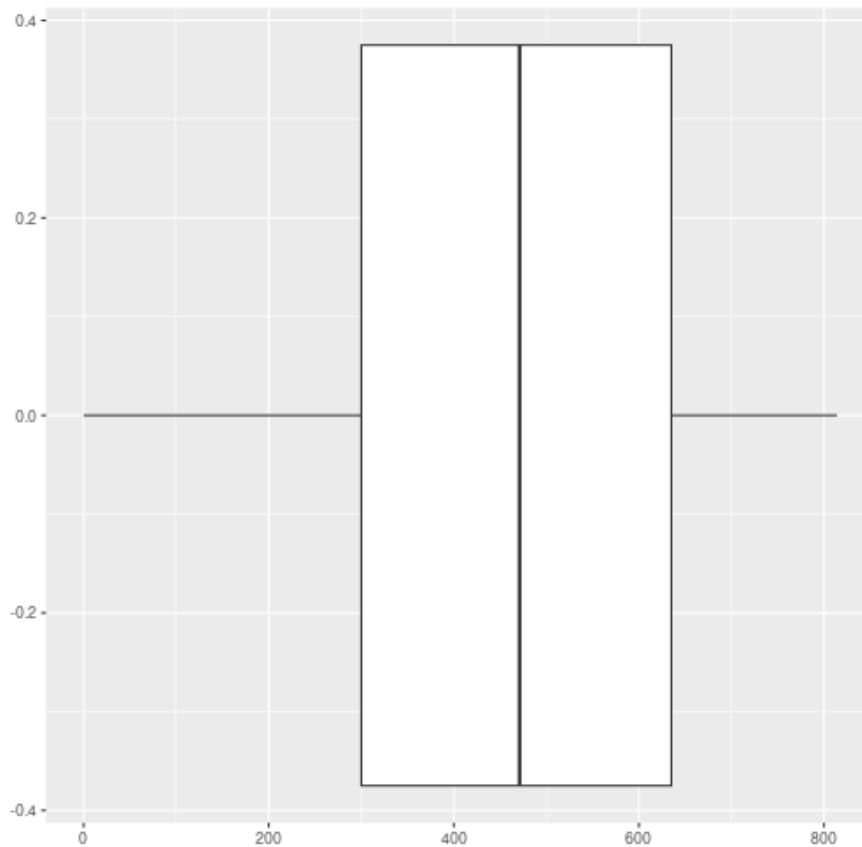
# Precio medio en Barra y Copacabana

Otra herramienta grafica bastante interesante es el `boxplot`.

Se trata de una gráfico que, como el histograma, nos permite entender la distribución de los valores de una columna, pero además nos brindas otras informaciones importantes:

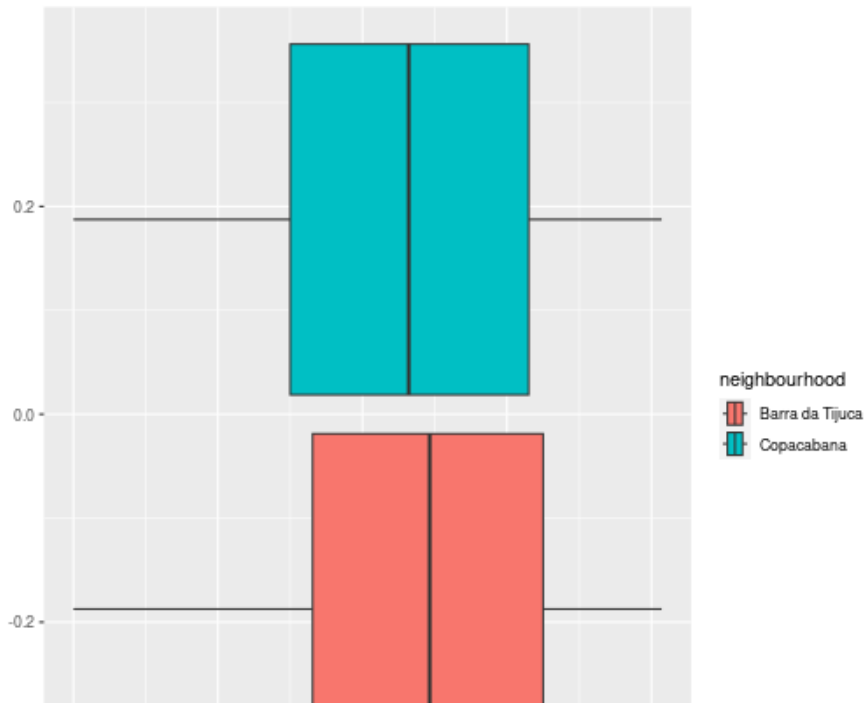
# variación de valores con boxplot

```
ggplot(airbnb_814) +  
  geom_boxplot(aes(x=price))
```



# boxplot por barrio

```
ggplot(airbnb_814) +  
  geom_boxplot(  
    aes(x=price,  
        group=neighbourhood,  
        fill=neighbourhood))
```



# Dividiendo el grafico por barrio

La función `facet_grid()` de `ggplot2`, nos permite separar los graficos por una determinada variable. Ya tenemos nuestro boxplot diferenciando los valores acorde al barrio (`neighbourhood`). Con el `facet_grid` podremos dividirlos también por el tipo de alquiler...

```
ggplot(airbnb_814) +  
  geom_boxplot(  
    aes(x=price,  
        group=neighbourhood,  
        fill=neighbourhood)) +  
  facet_grid(.~room_type)
```

# Dividiendo el grafico por barrio

