

---

# **VISUALIZACION DE DATOS EN RSTUDIO**

**EDUARD LARA**

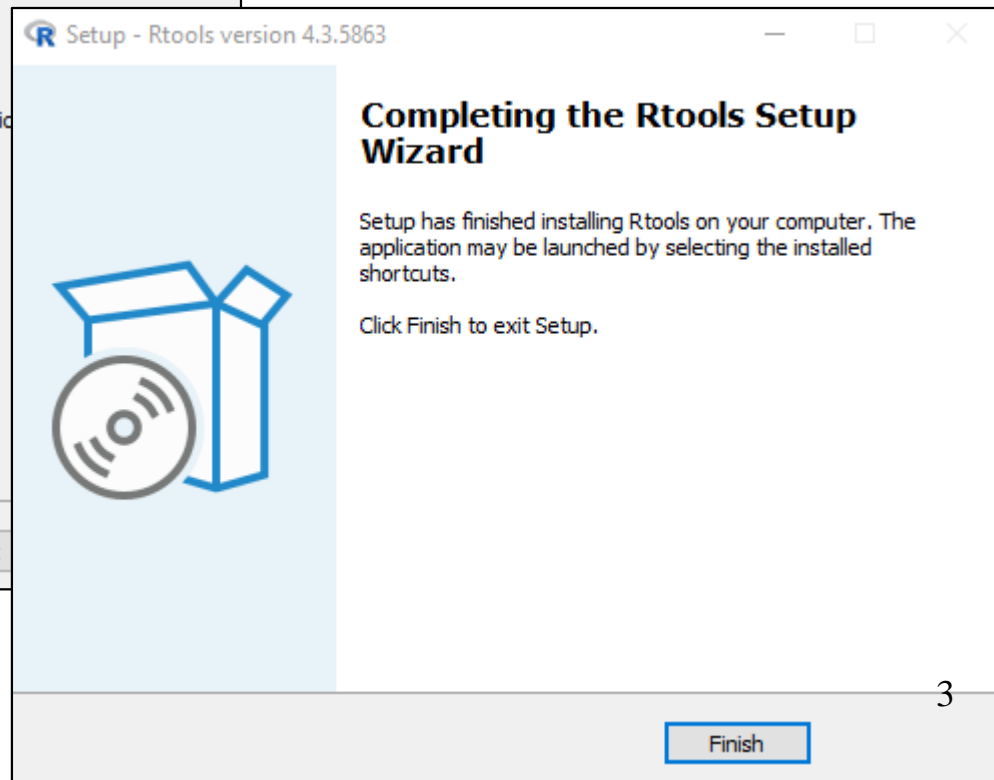
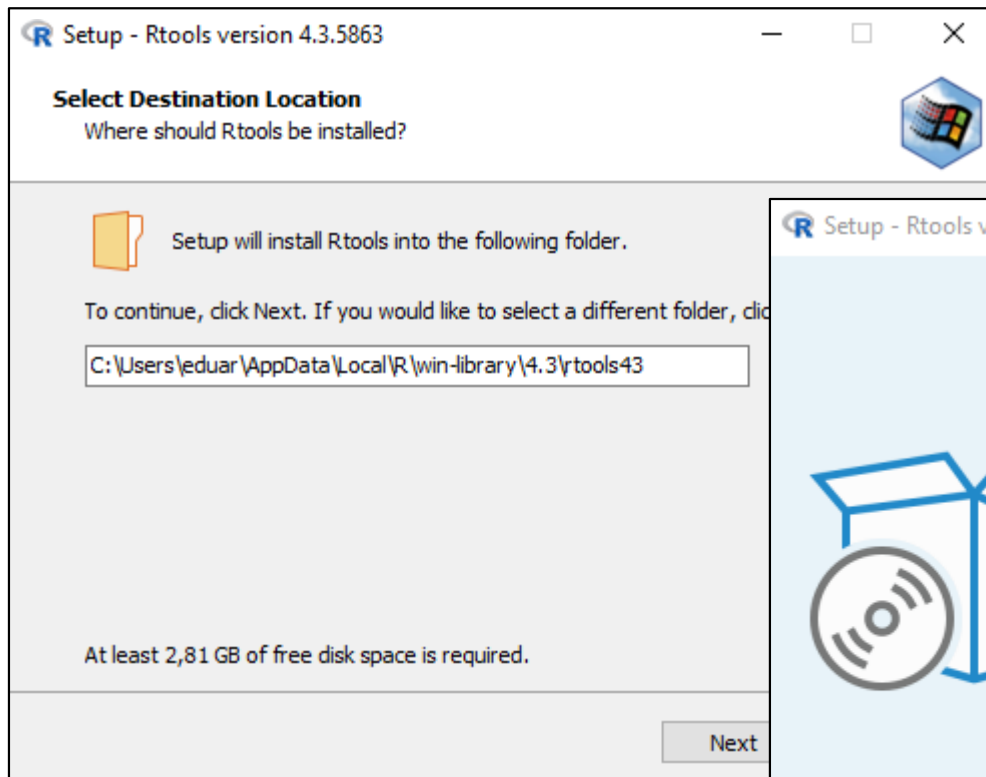
# INDICE

---

1. Histogramas
2. ScatterPlots
3. Barplots
4. Boxplots
5. Gráficos para la distribución de 2 variables
6. Limites y dimensiones de los gráficos
7. Gráficos interactivos con plotly

# 1. HISTOGRAMAS

**Paso 1.** Instalaremos la herramienta Rtools  
<https://cran.rstudio.com/bin/windows/Rtools/>



# 1. HISTOGRAMAS

**Paso 2.** Los histogramas son gráficos de una única variable con valores continuos en el tiempo. Para explicar los histogramas, usaremos dos paquetes que vamos a instalar:

- `ggplot2`, paquete visualización más importante de datos
- `ggplot2movies`, contiene un data set con información de películas

```
Console Terminal Background Jobs
R 4.3.2 ~./workspace/
> install.packages('ggplot2')
Installing package into 'C:/Users/eduar/AppData/Local/R/win-library/4.3'
(as 'lib' is unspecified)
probando la URL 'https://cran.rstudio.com/bin/windows/contrib/4.3/ggplot2_3.4.4.zip'
Content type 'application/zip' length 4299768 bytes (4.1 MB)
downloaded 4.1 MB

package 'ggplot2' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
C:\Users\eduar\AppData\Local\Temp\Rtmp80ZQHI\downloaded_packages
> install.packages('ggplot2movies')
Installing package into 'C:/Users/eduar/AppData/Local/R/win-library/4.3'
(as 'lib' is unspecified)
probando la URL 'https://cran.rstudio.com/bin/windows/contrib/4.3/ggplot2movies_0.0.1.zip'
Content type 'application/zip' length 1250961 bytes (1.2 MB)
downloaded 1.2 MB

package 'ggplot2movies' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
C:\Users\eduar\AppData\Local\Temp\Rtmp80ZQHI\downloaded_packages
> |
```

# 1. HISTOGRAMAS

---

**Paso 3.** Utilizamos la función `library` para cargar en memoria estos dos paquetes.

```
> library(ggplot2)
want to understand how all the pieces fit together? Read R for Data Science:
https://r4ds.had.co.nz/
> library(ggplot2movies)
> |
```

# 1. HISTOGRAMAS


Paso 4. Una documentación bastante útil para ggplot2, la encontramos en esta ruta:

[https://diegokoz.github.io/intro\\_ds/fuentes/ggplot2-cheatsheet-2.1-Spanish.pdf](https://diegokoz.github.io/intro_ds/fuentes/ggplot2-cheatsheet-2.1-Spanish.pdf)

donde aparecen ejemplos de todos los gráficos que podemos hacer

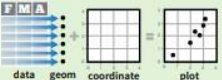
### Visualización de Datos usando ggplot2

Guía Rápida

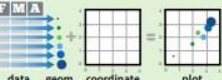


#### Conceptos Básicos

ggplot2 se basa en la idea que cualquier gráfica se puede construir usando estos tres componentes: **datos**, **coordenadas** y **objetos geométricos (geoms)**. Este concepto se llama: **gramática de las gráficas**.



Para visualizar resultados, asigne variables a las propiedades visuales, o **estéticas**, como **tamaño**, **color** y **posición** x ó y.



### Geoms

- Funciones geom se utilizan para visualizar resultados. Asigne variables a las propiedades estéticas del geom. Cada geom forma una capa.

#### Geométricas Elementales

```
a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))
```

- a + geom\_blank()**  
(Bueno para expandir límites)
- b + geom\_curve()**  
`aes(yend = lat + 1, xend=long+1, curvature=z)` - x, xend, y, yend, alpha, angle, color, curvature, linetype, size
- a + geom\_path()**  
`lineend="butt", linejoin="round", linemitre=1` - x, y, alpha, color, group, linetype, size
- a + geom\_polygon()**  
`aes(group = group)` - x, y, alpha, color, fill, group, linetype, size
- b + geom\_rect()**  
`aes(xmin = long, ymin=lat, xmax=long+1, ymax=lat+1)` - x, xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size
- a + geom\_ribbon()**  
`aes(ymin=unemploy - 900, ymax=unemploy + 900)` - x, ymax, ymin, alpha, color, fill, group, linetype, size

#### Segmentos Lineales

```
b + geom_abline(aes(intercept=0, slope=1))
b + geom_hline(aes(yintercept = lat))
```

#### Dos Variables

##### X Continua, Y Continua

```
e <- ggplot(mpg, aes(cty, hwy))
```

- e + geom\_label()**  
`aes(label = cty)`, `nudge_x = 1`, `nudge_y = 1`, `check_overlap = TRUE` - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust
- e + geom\_jitter()**  
`height = 2`, `width = 2` - x, y, alpha, color, fill, shape, size
- e + geom\_point()**  
- x, y, alpha, color, fill, shape, size, stroke
- e + geom\_quantile()**  
- x, y, alpha, color, group, linetype, size, weight
- e + geom\_rug()**  
`sides = "bl"` - x, y, alpha, color, linetype, size
- e + geom\_smooth()**  
`method = lm` - x, y, alpha, color, fill, group, linetype, size, weight
- e + geom\_text()**  
`aes(label = cty)`, `nudge_x = 1`, `nudge_y = 1`, `check_overlap = TRUE` - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

##### Distribución Bivariada Continua

```
h <- ggplot(diamonds, aes(carat, price))
```

- h + geom\_bin2d()**  
- x, y, alpha, color, fill, linetype, size, weight
- h + geom\_density2d()**  
- x, y, alpha, colour, group, linetype, size
- h + geom\_hex()**  
- x, y, alpha, colour, fill, size

##### Función Continua

```
i <- ggplot(economics, aes(date, unemploy))
```

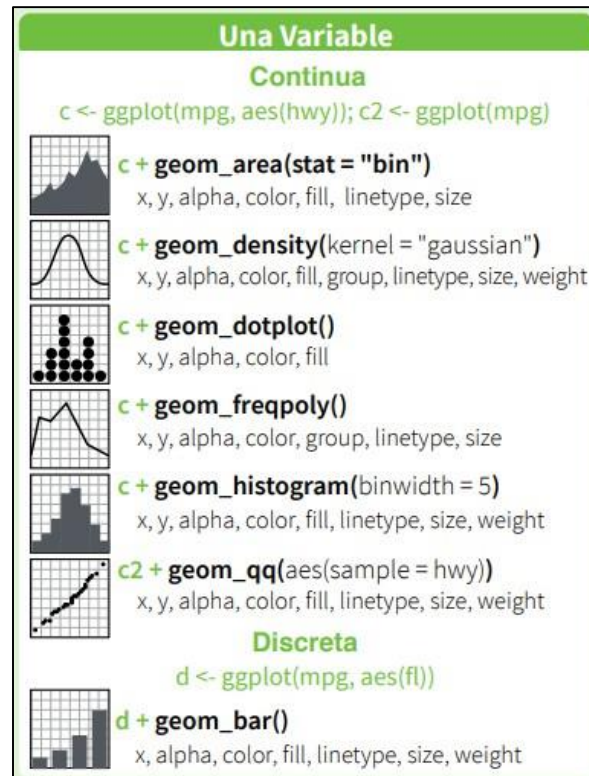
- i + geom\_area()**  
- x, y, alpha, color, fill, linetype, size
- i + geom\_line()**  
- x, y, alpha, color, group, linetype, size
- i + geom\_step()**  
`direction = "hv"` - x, y, alpha, color, group, linetype, size

##### Visualizando el Error

```
df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
i <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))
```

# 1. HISTOGRAMAS

**Paso 5.** Vamos a utilizar el gráfico `geom_histogram`, que necesita de una única variable, continua en el tiempo, y el diagrama de barras es un histograma. Podemos utilizar todos estos atributos para configurar nuestro gráfico.



# 1. HISTOGRAMAS

**Paso 6.** En Rstudio, primero creamos una variable películas que vamos a crear con los datos del data set movies que viene en el paquete ggplot2movies

Si hacemos un head de películas para ver las primeras líneas de este dataset, aquí tenemos el título, el año, la longitud, el rating, etc. Hay 9 variables más que están abajo

```
> películas=movies
> head(películas)
# A tibble: 6 x 24
  title    year length budget rating votes   r1    r2    r3    r4    r5    r6    r7    r8    r9
  <chr>   <int>   <int>   <int>   <dbl> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 $      1971    121    NA     6.4   348   4.5   4.5   4.5   4.5   14.5  24.5  24.5  14.5  4.5
2 $100... 1939     71    NA     6     20    0    14.5  4.5   24.5  14.5  14.5  14.5  4.5   4.5
3 $21 ... 1941      7    NA     8.2    5    0     0     0     0     24.5  0    44.5  24.5
4 $40,... 1996     70    NA     8.2    6   14.5    0     0     0     0     0     0     0    34.5
5 $50,... 1975     71    NA     3.4   17   24.5   4.5    0    14.5  14.5   4.5    0     0     0
6 $pent   2000     91    NA     4.3   45   4.5   4.5   4.5  14.5  14.5  14.5   4.5   4.5  14.5
# 9 more variables: r10 <dbl>, mpaa <chr>, Action <int>, Animation <int>, Comedy <int>,
#   Drama <int>, Documentary <int>, Romance <int>, Short <int>
> |
```



# 1. HISTOGRAMAS

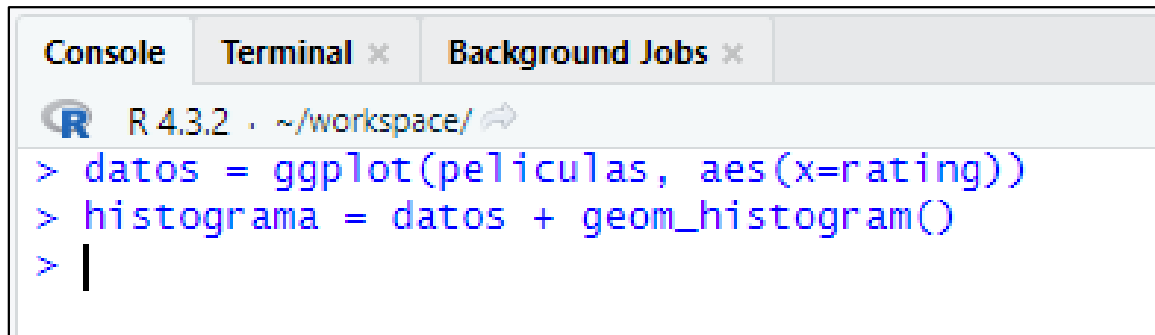
**Paso 7.** Para ver mejor el título, se puede hacer una selección de algunas columnas: título, año y rating. Así se ve mejor el título de las películas, el año y la puntuación

```
Console Terminal x Background Jobs x
R 4.3.2 · ~/workspace/ ↗
> películas[c('title','year','rating')]
# A tibble: 58,788 × 3
  title          year rating
  <chr>         <int> <dbl>
1 $             1971    6.4
2 $1000 a Touchdown 1939    6
3 $21 a Day Once a Month 1941    8.2
4 $40,000          1996    8.2
5 $50,000 climax Show, The 1975    3.4
6 $pent           2000    4.3
7 $windle          2002    5.3
8 '15'             2002    6.7
9 '38              1987    6.6
10 '49-'17         1917    6
# i 58,778 more rows
# i Use `print(n = ...)` to see more rows
> |
```

# 1. HISTOGRAMAS

**Paso 8.** Para crear el histograma, creamos una variable `datos` que van a ser los datos que vamos a poner en el histograma. Mediante `ggplot` pasaremos el dataset `películas` e indicamos que el valor en el eje X va a ser la columna `rating`.

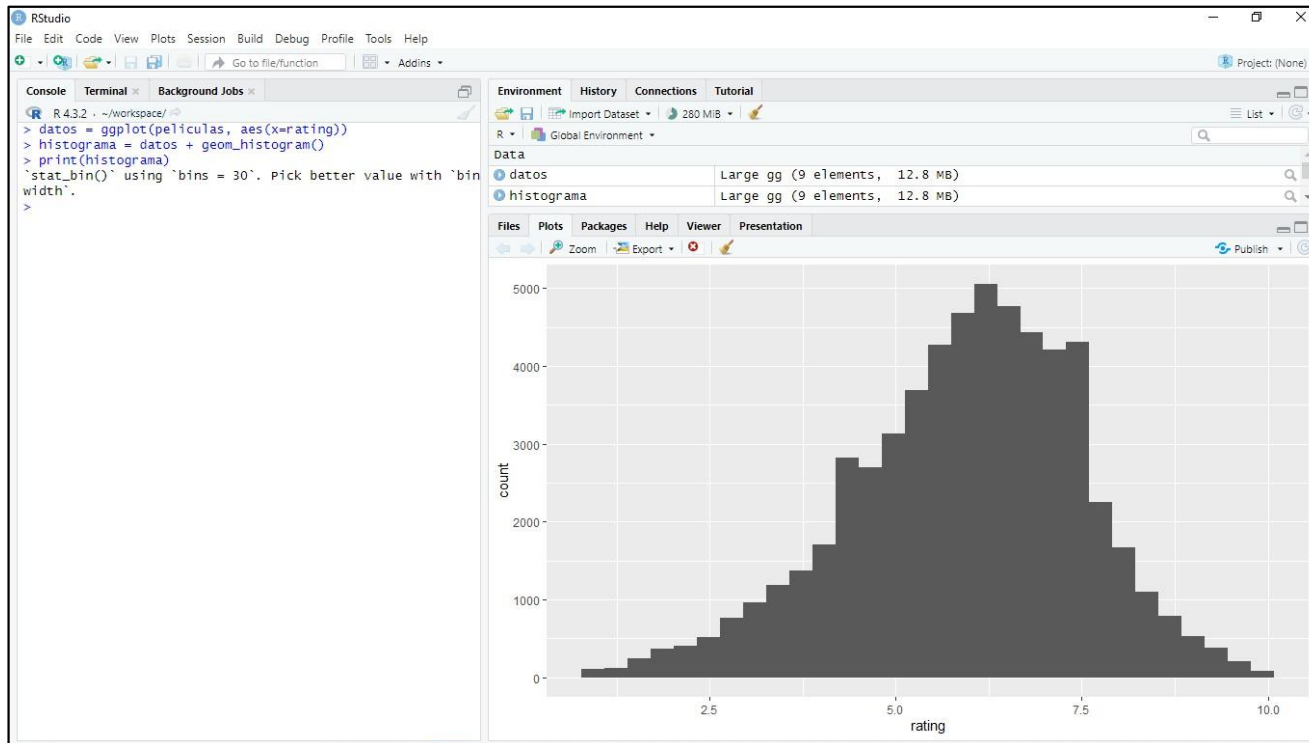
Creamos otra variable `histograma` que sean estos datos más `geom_histogram()`, para crear un histograma con una variable continua, sin ningún parámetro

A screenshot of an R console window. The window has three tabs at the top: 'Console', 'Terminal x', and 'Background Jobs x'. The 'Console' tab is active. Below the tabs, the R logo is followed by the text 'R 4.3.2 · ~/workspace/'. The console shows three lines of R code: the first line is '> datos = ggplot(películas, aes(x=rating))', the second line is '> histograma = datos + geom\_histogram()', and the third line is '> |' with a vertical cursor at the end.

```
R 4.3.2 · ~/workspace/
> datos = ggplot(películas, aes(x=rating))
> histograma = datos + geom_histogram()
> |
```

# 1. HISTOGRAMAS

**Paso 9.** Hacemos `print(histograma)`, para mostrar el gráfico. Inmediatamente se abre la pestaña pilots, donde aparece el histograma de la variable rating. Aparece el nº de veces que se repite cada uno de los valores de la variable.

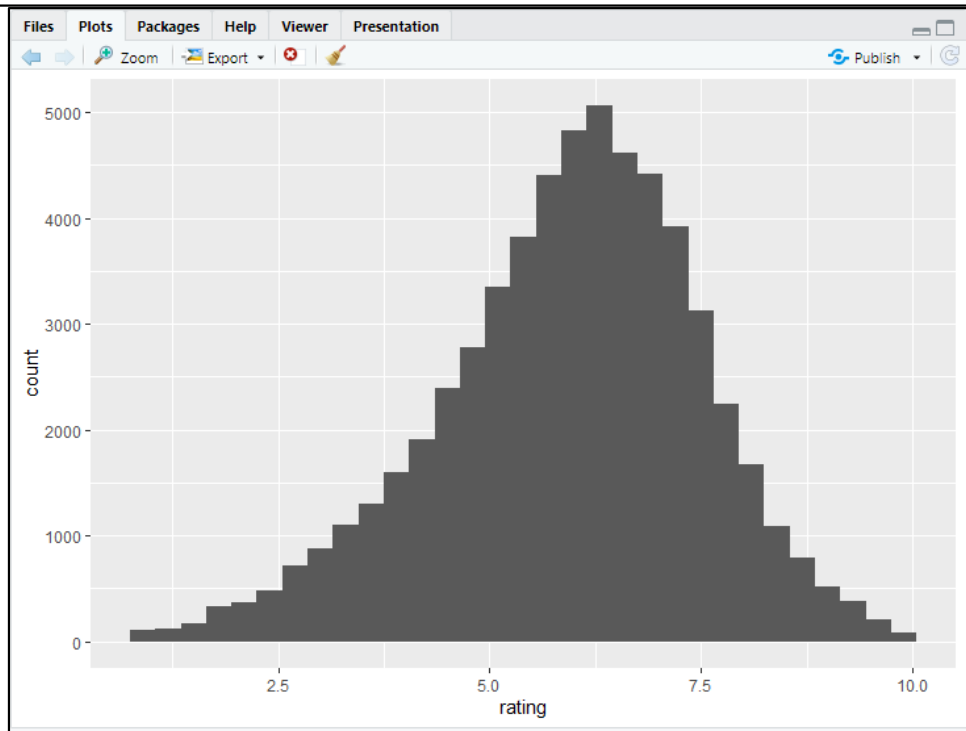


El rating 5 se repetiría unas 3 mil y pico veces.

# 1. HISTOGRAMAS

**Paso 10.** Podemos configurar el histograma para que tenga otra configuración. Por ejemplo podemos hacer que las columnas sean un poco más estrechas que la anterior a 0.3.

```
> histograma = datos + geom_histogram(binwidth = 0.3)
> print(histograma)
> |
```

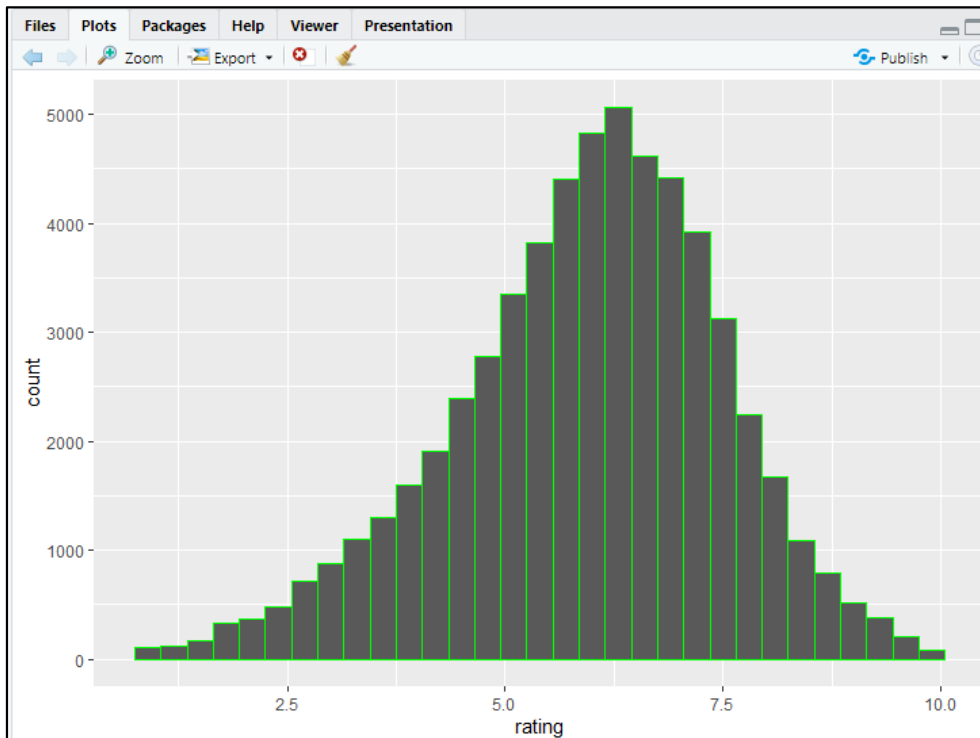


Con el botón flecha debajo de Files, podemos ir al gráfico anterior y volver al nuevo para comprobar que se ha estrechado el ancho de estas columnas

# 1. HISTOGRAMAS

**Paso 11.** También podemos cambiar el color, por ejemplo de color verde.

```
> histograma = datos + geom_histogram(binwidth = 0.3, color='green')  
> print(histograma)  
>
```

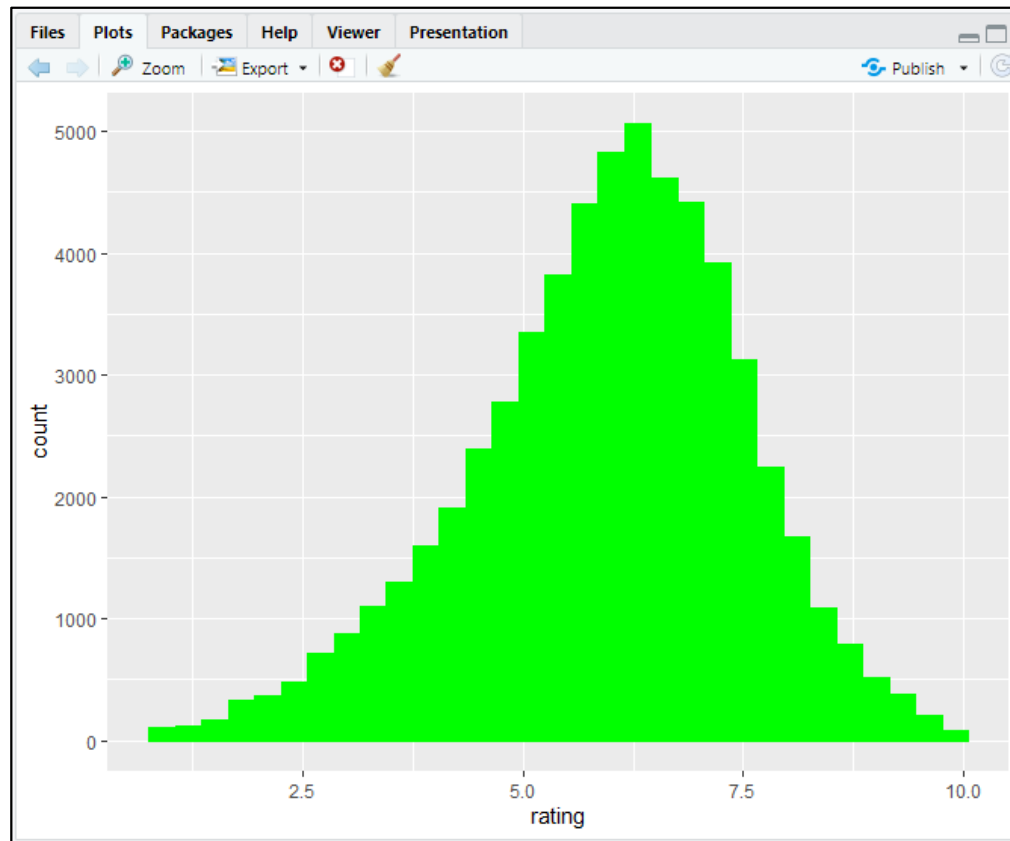


Aparece el contorno de color verde y el interior lo ha dejado en el color que ya está.

# 1. HISTOGRAMAS

**Paso 12.** Podemos cambiar el color del interior con fill.  
Ponemos tanto el contorno como el relleno del gráfico verde

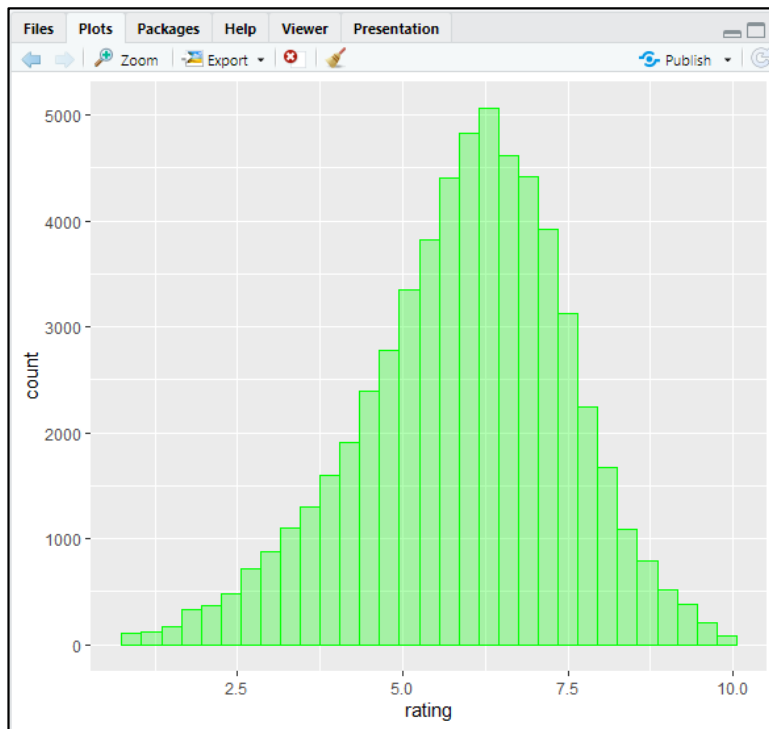
```
> histograma = datos + geom_histogram(binwidth = 0.3, color='green',fill='green')  
> print(histograma)  
> |
```



# 1. HISTOGRAMAS

**Paso 13.** Las líneas horizontales no se ven por detrás del gráfico. Podemos hacerlo un poco transparente, con el atributo `alpha`, para que se vean las líneas que pasan por detrás.

```
> histograma = datos + geom_histogram(binwidth = 0.3, color='green', fill='green', alpha=0.3)
> print(histograma)
> |
```

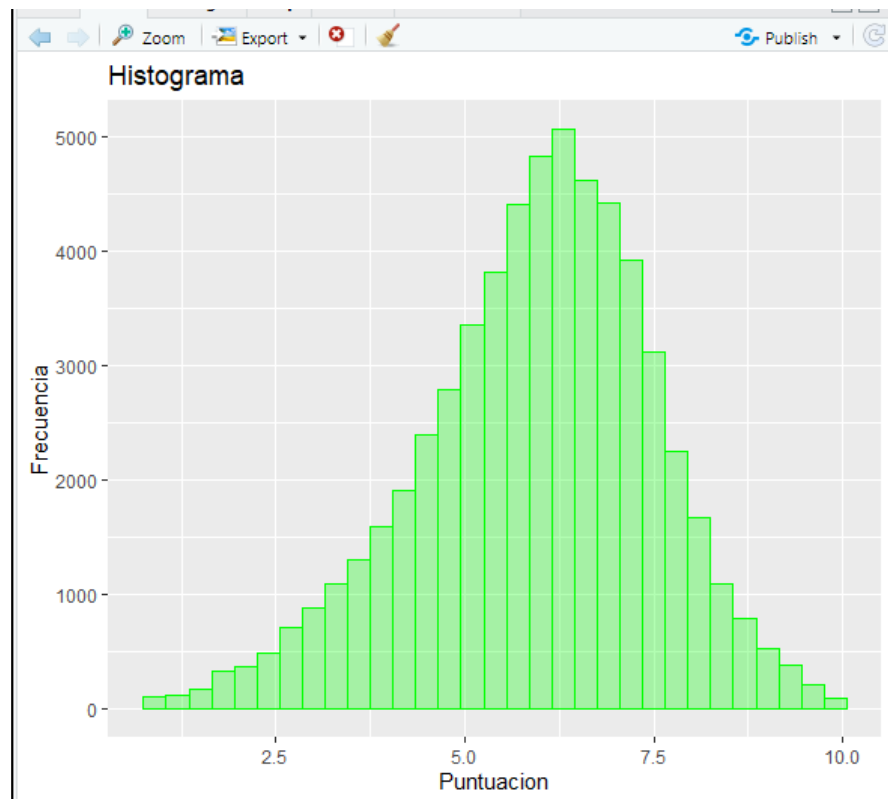


Vemos que se ha suavizado un poco el color del relleno para que se vean las líneas con los números de la frecuencia

# 1. HISTOGRAMAS

**Paso 14.** También podemos cambiarle el nombre del eje de las X, el eje de las y ponerle un título al gráfico.

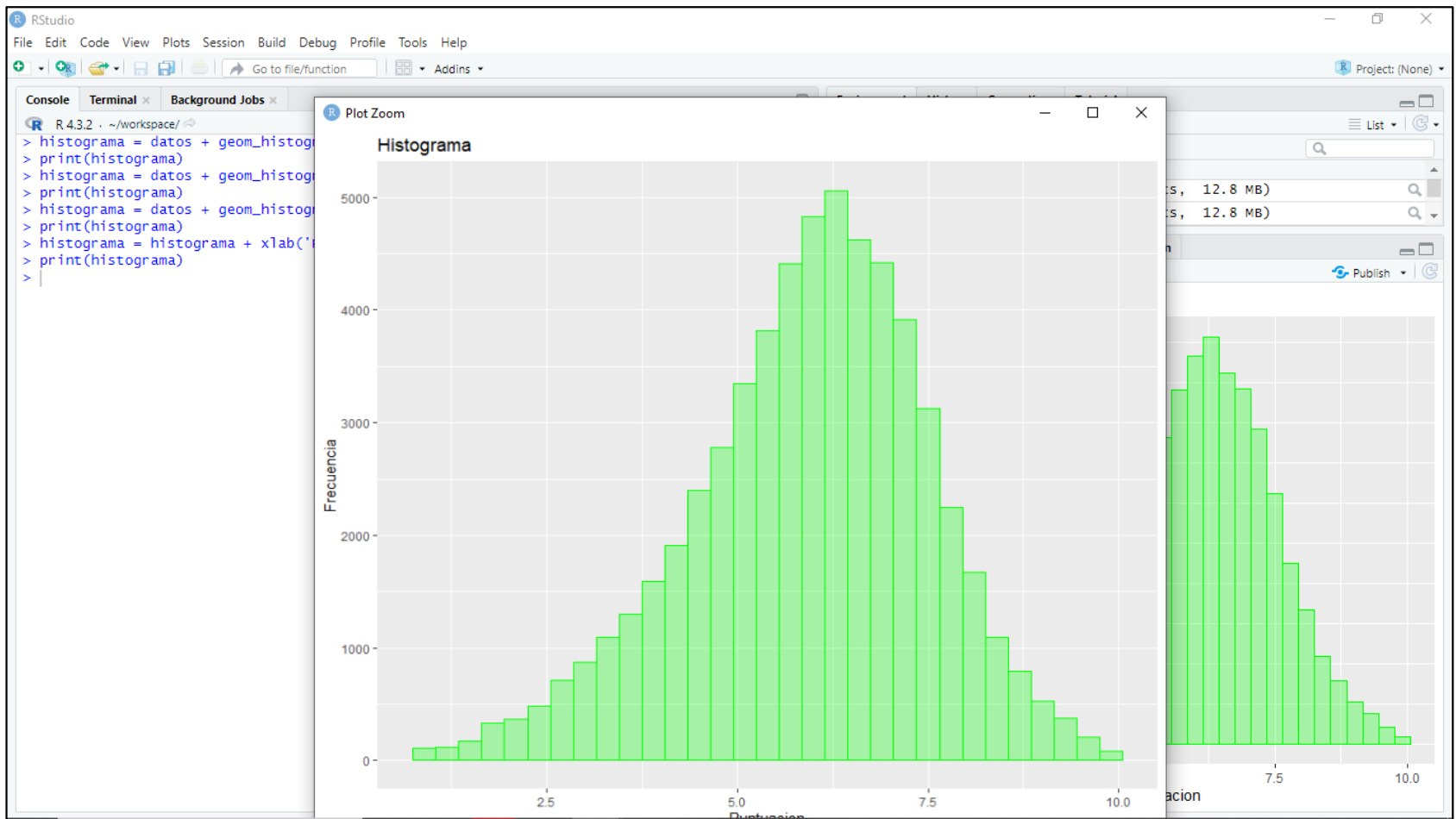
```
> histograma = histograma + xlab('Puntuacion') + ylab('Frecuencia') + ggtitle('Histograma')  
> print(histograma)  
>
```





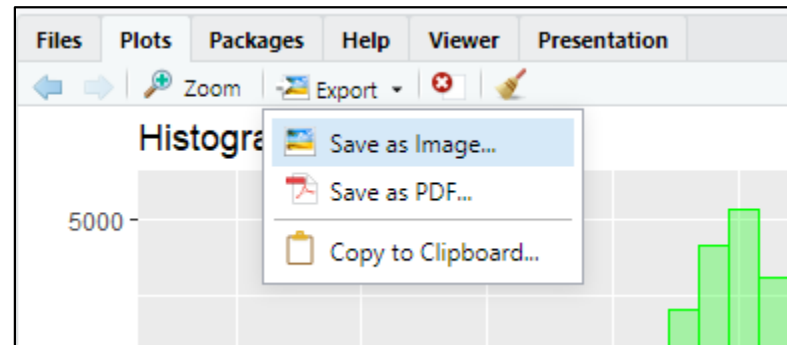
# 1. HISTOGRAMAS

**Paso 15.** Si le damos al botón de zoom, se abre una pantalla grande donde podemos ver mejor el gráfico



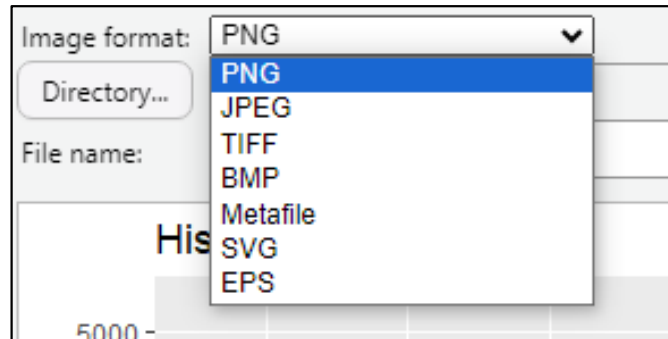
# 1. HISTOGRAMAS

**Paso 16.** Mediante el botón export podemos exportar el gráfico como su propia imagen, como si fuera un PDF o al portapapeles para pegarlo en otra aplicación

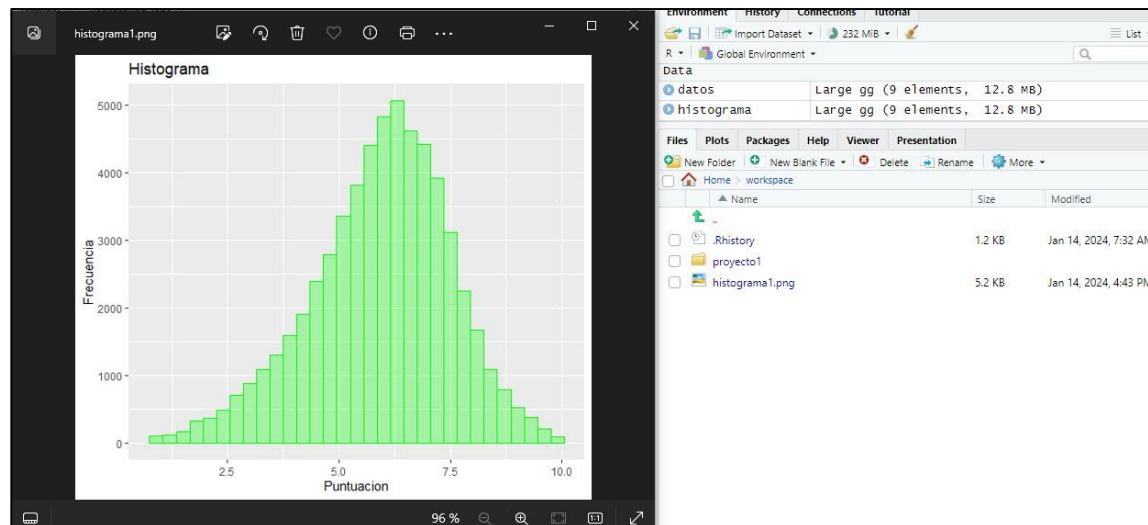


# 1. HISTOGRAMAS

**Paso 17.** Si seleccionamos exportar como imagen, lo guardaremos dentro de nuestro workspace en diferentes formatos

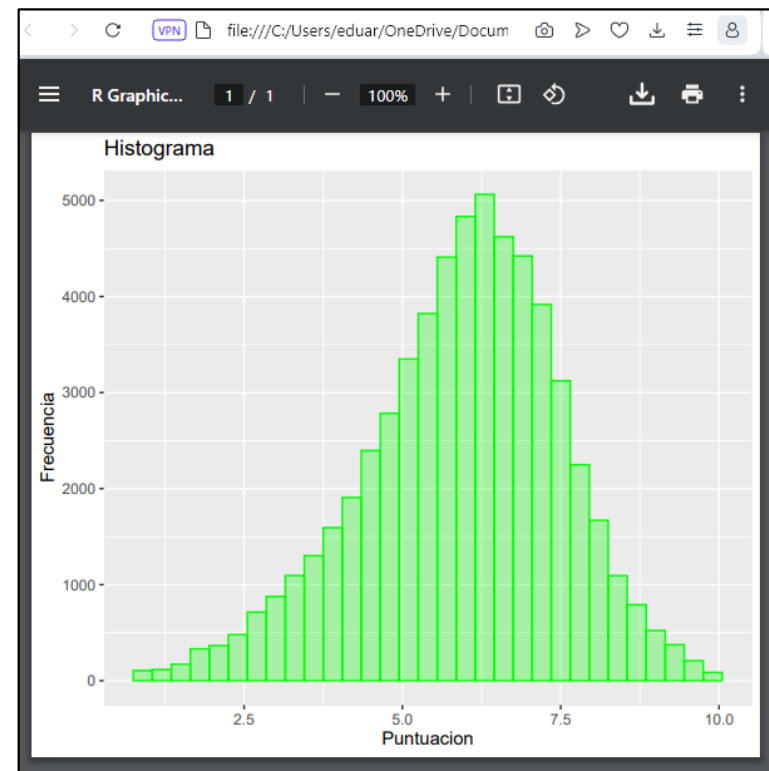
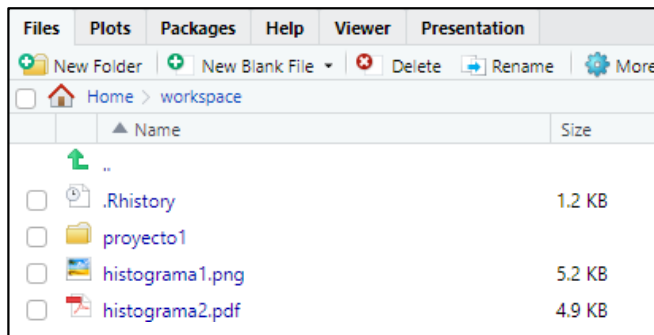
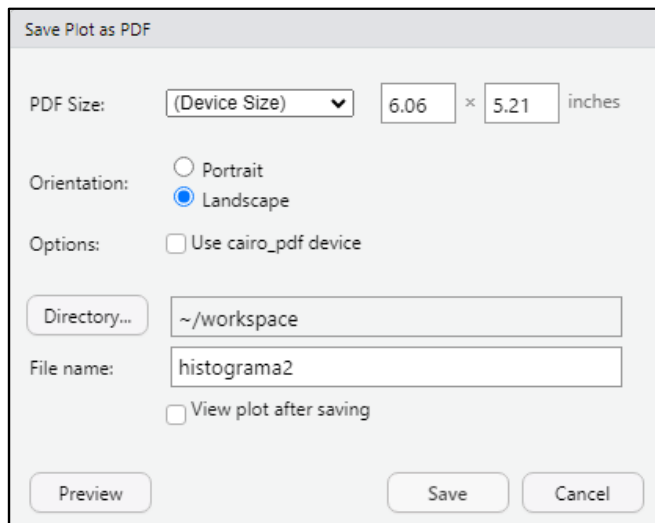


Si lo pulsamos en Files nos abrirá el fichero



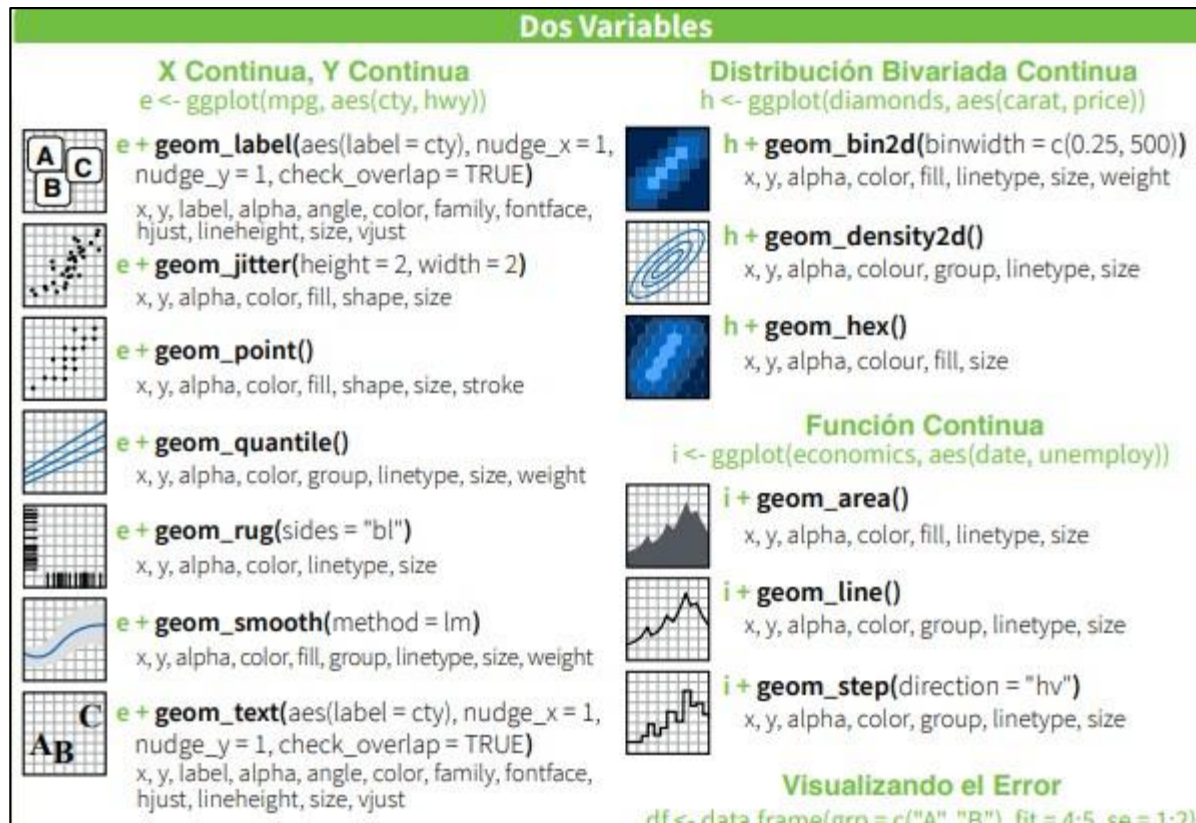
# 1. HISTOGRAMAS

**Paso 18.** También lo podemos exportar como PDF. Si vamos a Files nos aparece como fichero pdf. Si hacemos click nos abrirá el PDF en el navegador



## 2. SCATTERPLOTS

**Paso 1.** Los Scatterplots son gráficos con dos variables continuas. Si vamos a la pagina de ayuda del paquete ggplot2, seleccionaremos scatterplot mediante la función `geom_point()`



## 2. SCATTERPLOTS

---

**Paso 2.** Primero cargamos ggplot2 con library. Despues creamos una variable coches que va a contener la información del mtcars que viene como ejemplo en Rstudio  
Contiene información de los coches en las diferentes columnas del dataset

```
> library(ggplot2)
> coches = mtcars
> head(coches)
```

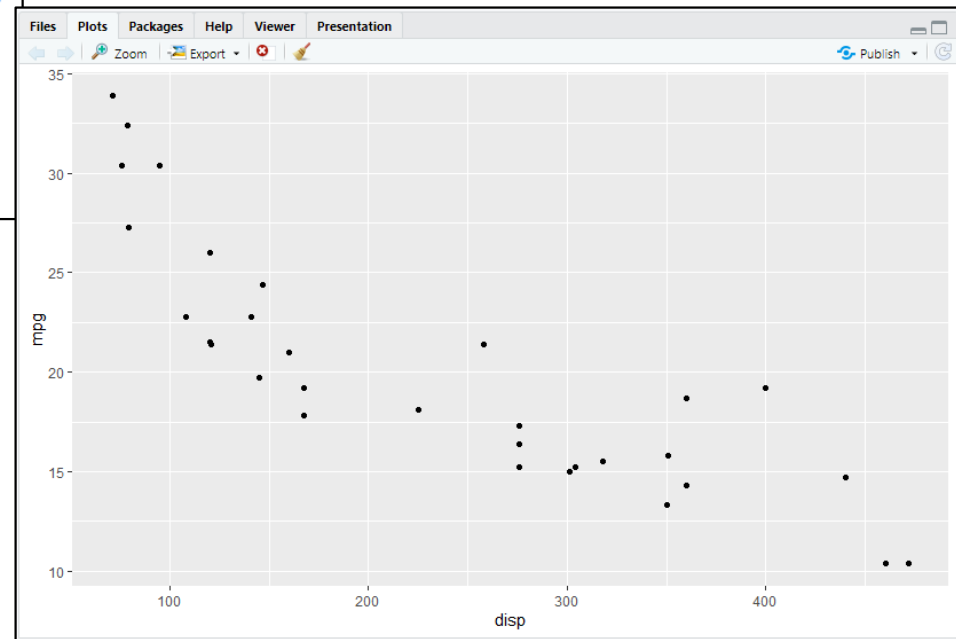
|                   | mpg  | cyl | disp | hp  | drat | wt    | qsec  | vs | am | gear | carb |
|-------------------|------|-----|------|-----|------|-------|-------|----|----|------|------|
| Mazda RX4         | 21.0 | 6   | 160  | 110 | 3.90 | 2.620 | 16.46 | 0  | 1  | 4    | 4    |
| Mazda RX4 Wag     | 21.0 | 6   | 160  | 110 | 3.90 | 2.875 | 17.02 | 0  | 1  | 4    | 4    |
| Datsun 710        | 22.8 | 4   | 108  | 93  | 3.85 | 2.320 | 18.61 | 1  | 1  | 4    | 1    |
| Hornet 4 Drive    | 21.4 | 6   | 258  | 110 | 3.08 | 3.215 | 19.44 | 1  | 0  | 3    | 1    |
| Hornet Sportabout | 18.7 | 8   | 360  | 175 | 3.15 | 3.440 | 17.02 | 0  | 0  | 3    | 2    |
| valiant           | 18.1 | 6   | 225  | 105 | 2.76 | 3.460 | 20.22 | 1  | 0  | 3    | 1    |

## 2. SCATTERPLOTS

**Paso 3.** Entonces vamos a crear un scatterplot, donde pondremos en el eje X la columna disp y en el eje y la columna mpg. Añadiremos la función `geom_point` e imprimimos el gráfico.

```
Console Terminal x Background Jobs x
R 4.3.2 . ~/workspace/
> grafico = ggplot(coches, aes(x=disp, y=mpg))
>
> grafico = grafico + geom_point()
>
> print(grafico)
> |
```

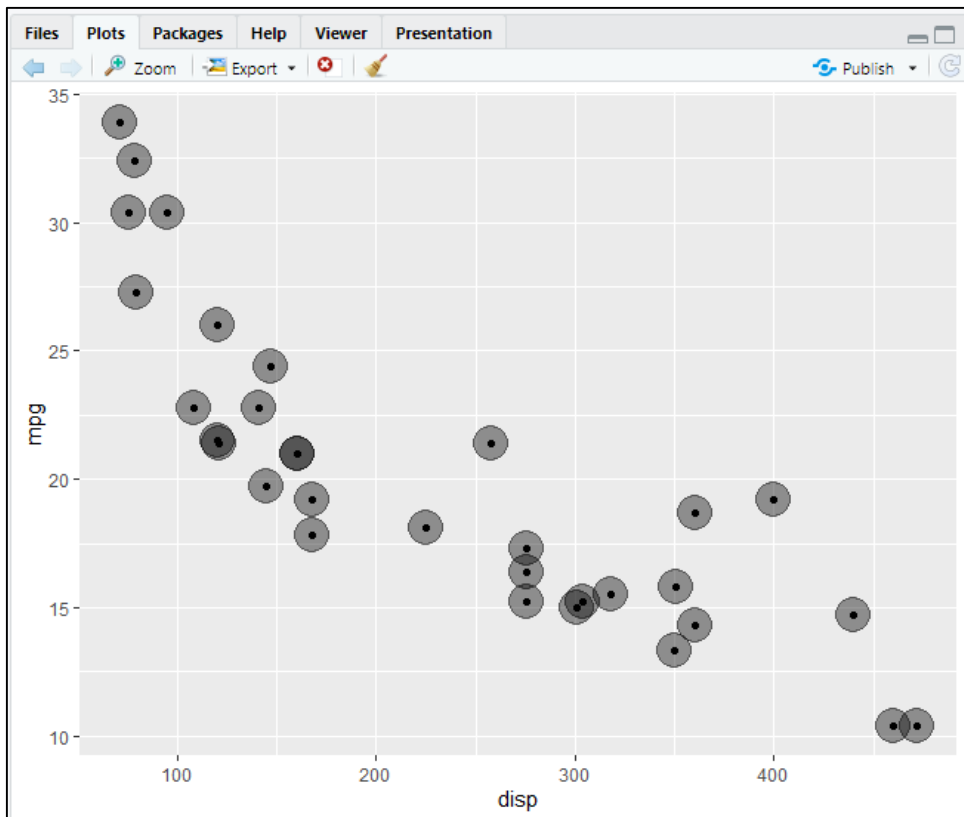
Mediante un scatterplot hemos relacionado las dos variables, la columna disp, y la columna mpg (eje y)



## 2. SCATTERPLOTS

**Paso 4.** Podemos cambiar el tamaño de los puntos. Ponemos un tamaño de 8, y una transparencia del 40%.

```
> grafico = grafico + geom_point(size=8, alpha=0.4)
> print(grafico)
> |
```



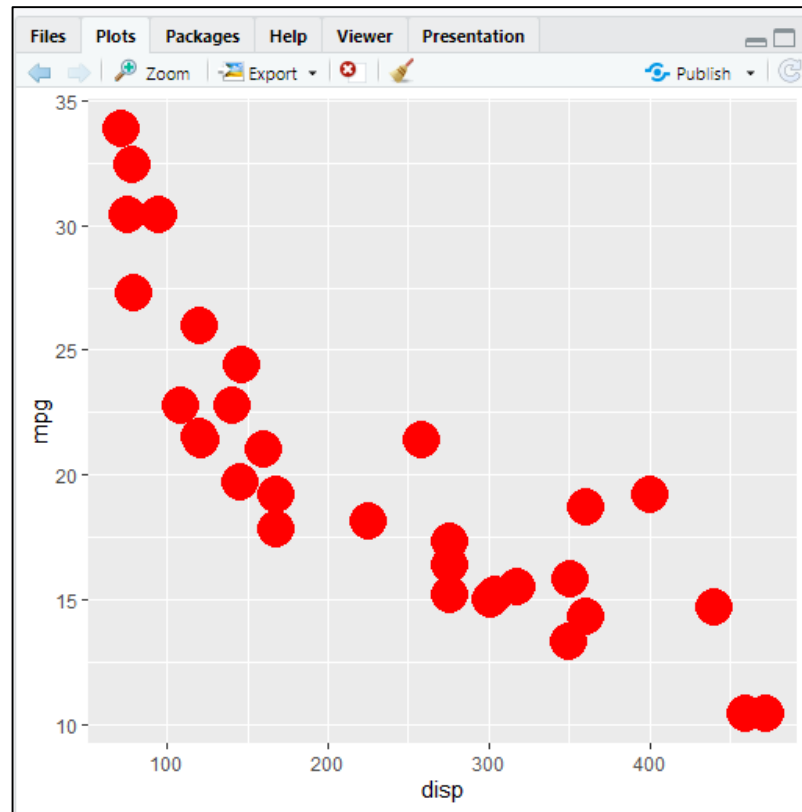
Ha aumentado el círculo al tamaño de 8 y ha hecho una transparencia, donde se ve más clarito el gris. Así permite ver dónde se junta con otros puntos.



## 2. SCATTERPLOTS

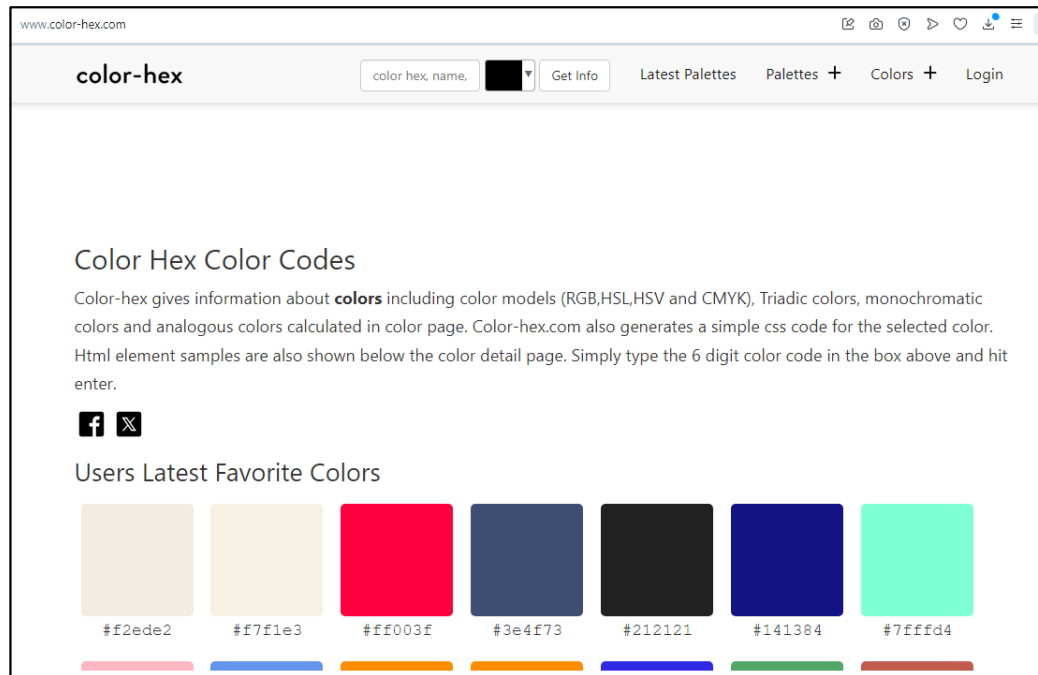
**Paso 5.** Ahora vamos a cambiar el color. En vez de gris le ponemos el color rojo.

```
> grafico = grafico + geom_point(size=8, color='red')  
> print(grafico)  
> |
```



## 2. SCATTERPLOTS

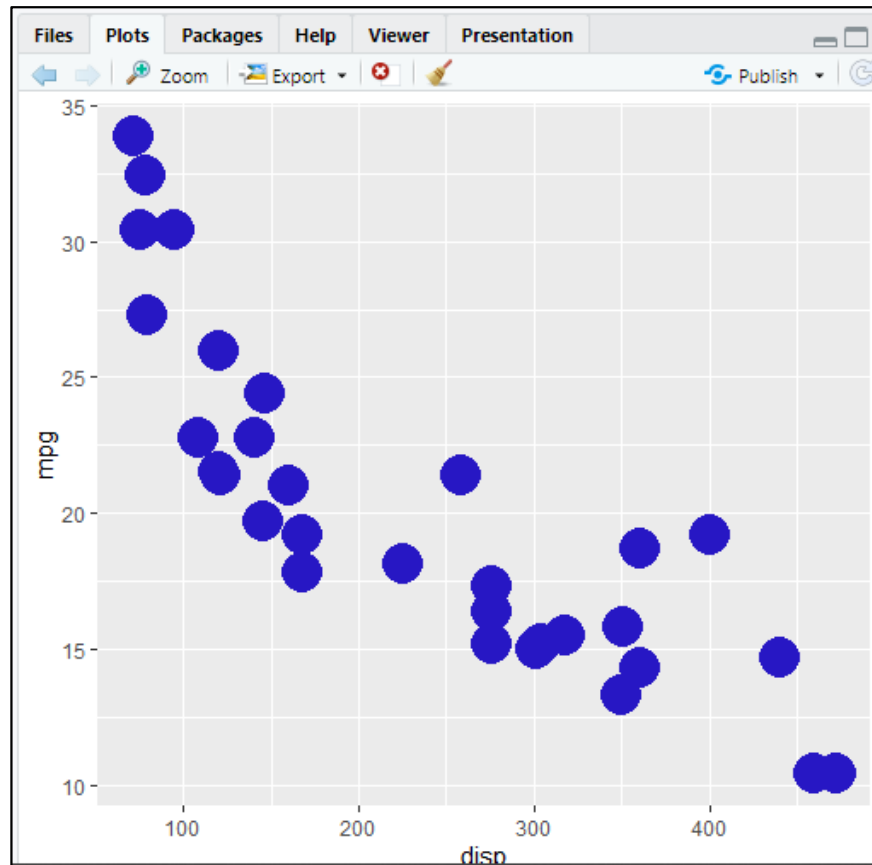
**Paso 6.** Existe una página en internet [www.color-hex.com](http://www.color-hex.com) donde podemos seleccionar diferentes tipos de colores. Cada color que podemos elegir lleva una almohadilla seguida de un código hexadecimal, que lo identifica. Los utilizaremos para crear nuestro gráfico, nuestros puntos del gráfico con ese color.



## 2. SCATTERPLOTS

**Paso 7.** Configuramos un color con su código hexadecimal, en el gráfico

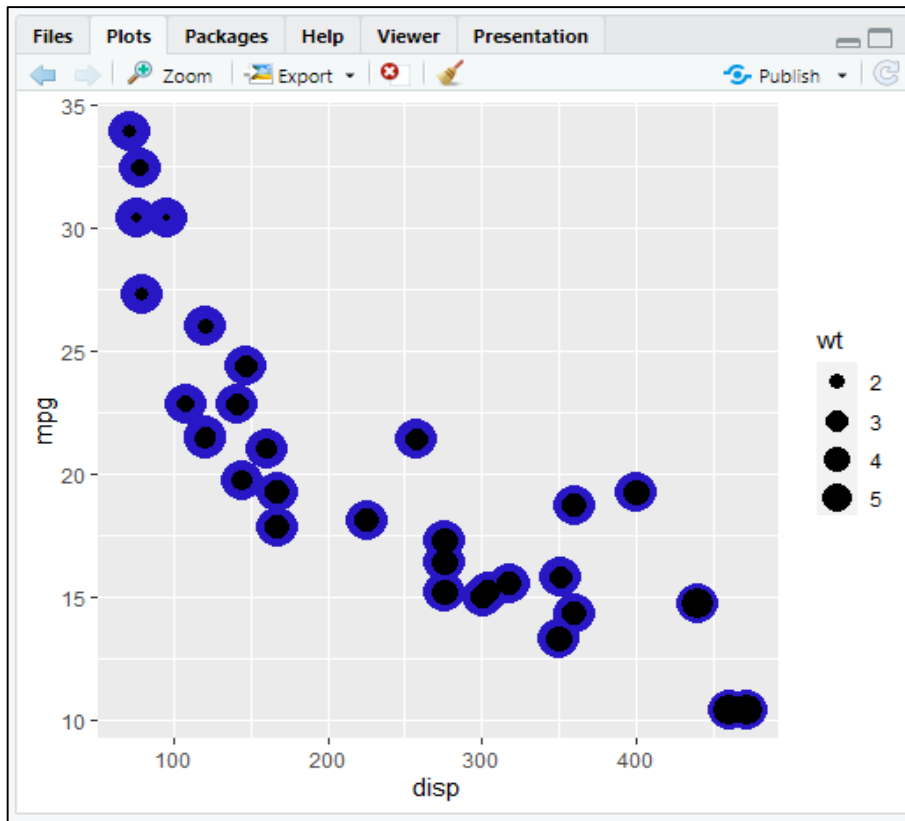
```
> grafico = grafico + geom_point(size=8, color='#2717c4')  
> print(grafico)  
>
```



## 2. SCATTERPLOTS

**Paso 8.** Podemos configurar también el tamaño del punto, y ponerlo en función del valor de otra columna. .

```
> grafico = grafico + geom_point(aes(size=wt))  
> print(grafico)  
> |
```



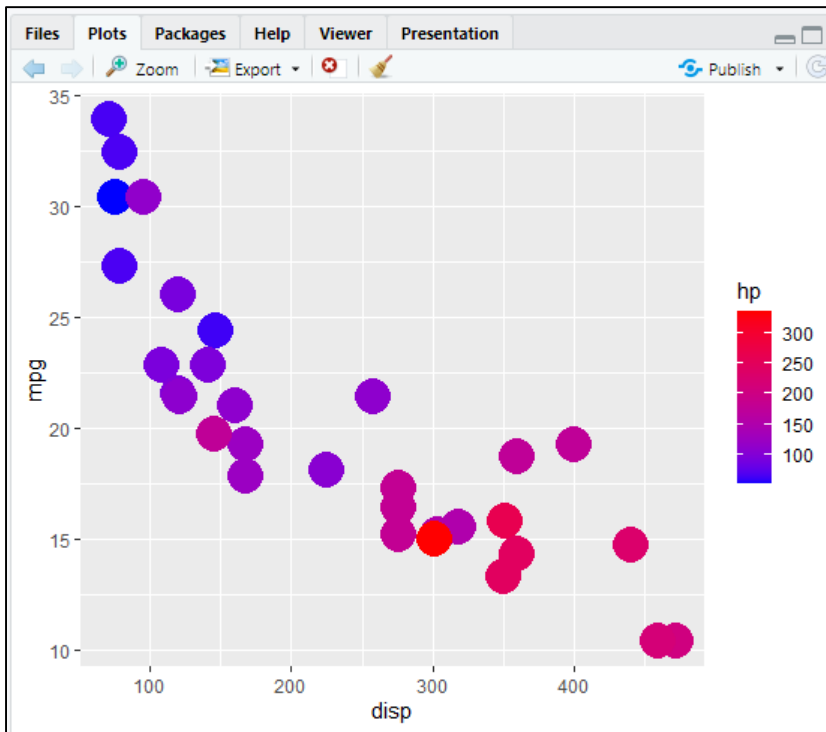
Va cambiando el tamaño del punto en función del valor de otra columna.

Es otra forma de hacerlo también para relacionar, en este caso tres columnas.

## 2. SCATTERPLOTS

**Paso 9.** Si queremos hacer un tipo de gráfico, como una especie de gradiente de colores

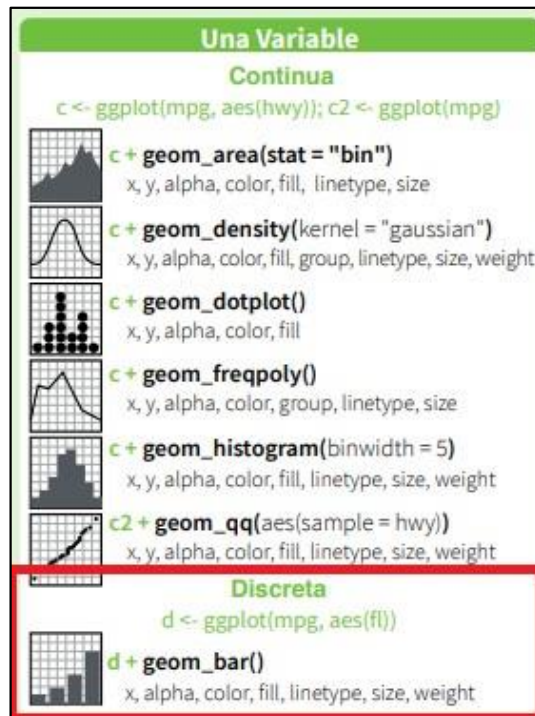
```
> grafico = ggplot(coches, aes(x=disp,y=mpg))  
> grafico = grafico + geom_point(size=8, aes(color=hp))  
> grafico = grafico + scale_color_gradient(low='blue',high='red')  
> print(grafico)  
>
```



De esta forma podemos combinar tres variables con una gráfica de colores gradientes en función de la columna HP.

### 3. BARPLOTS

**Paso 1.** Un barplot es un gráfico con una única variable discreta. Si vamos a la ayuda sobre el paquete de ggplot2, elegiremos la función `geom_bar()` que genera un diagrama de barras, donde el eje de las X pondremos una variable de tipo discreto con valores alfanuméricos etc



### 3. BARPLOTS

**Paso 2.** Cargamos la librería ggplot2 y crearemos una variable datos que contendrá la dataset mpg. Si visualizamos las primeras líneas de mpg veremos que tiene diferentes columnas a nivel de coches, el constructor, el modelo, el año, etc.

```
> library(ggplot2)
> datos = mpg
> head(datos)
# A tibble: 6 × 11
  manufacturer model displ  year   cyl trans      drv    cty   hwy fl    class
  <chr>         <chr> <dbl> <int> <int> <chr>    <chr> <int> <int> <chr> <chr>
1 audi         a4      1.8  1999     4 auto(l5) f      18    29 p    compact
2 audi         a4      1.8  1999     4 manual(m5) f      21    29 p    compact
3 audi         a4      2    2008     4 manual(m6) f      20    31 p    compact
4 audi         a4      2    2008     4 auto(av) f      21    30 p    compact
5 audi         a4      2.8  1999     6 auto(l5) f      16    26 p    compact
6 audi         a4      2.8  1999     6 manual(m5) f      18    26 p    compact
```

### 3. BARPLOTS

---

**Paso 3.** Si queremos ver la estructura de la base de datos, vemos que tiene 234 ocurrencias, con 11 columnas

```
> str(datos)
tibble [234 × 11] (S3: tbl_df/tbl/data.frame)
 $ manufacturer: chr [1:234] "audi" "audi" "audi" "audi" ...
 $ model       : chr [1:234] "a4" "a4" "a4" "a4" ...
 $ displ       : num [1:234] 1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...
 $ year        : int [1:234] 1999 1999 2008 2008 1999 1999 2008 1999 1999 2008 ...
 $ cyl         : int [1:234] 4 4 4 4 6 6 6 4 4 4 ...
 $ trans       : chr [1:234] "auto(l5)" "manual(m5)" "manual(m6)" "auto(av)" ...
 $ drv         : chr [1:234] "f" "f" "f" "f" ...
 $ cty         : int [1:234] 18 21 20 21 16 18 18 18 16 20 ...
 $ hwy         : int [1:234] 29 29 31 30 26 26 27 26 25 28 ...
 $ fl          : chr [1:234] "p" "p" "p" "p" ...
 $ class       : chr [1:234] "compact" "compact" "compact" "compact" ...
>
```

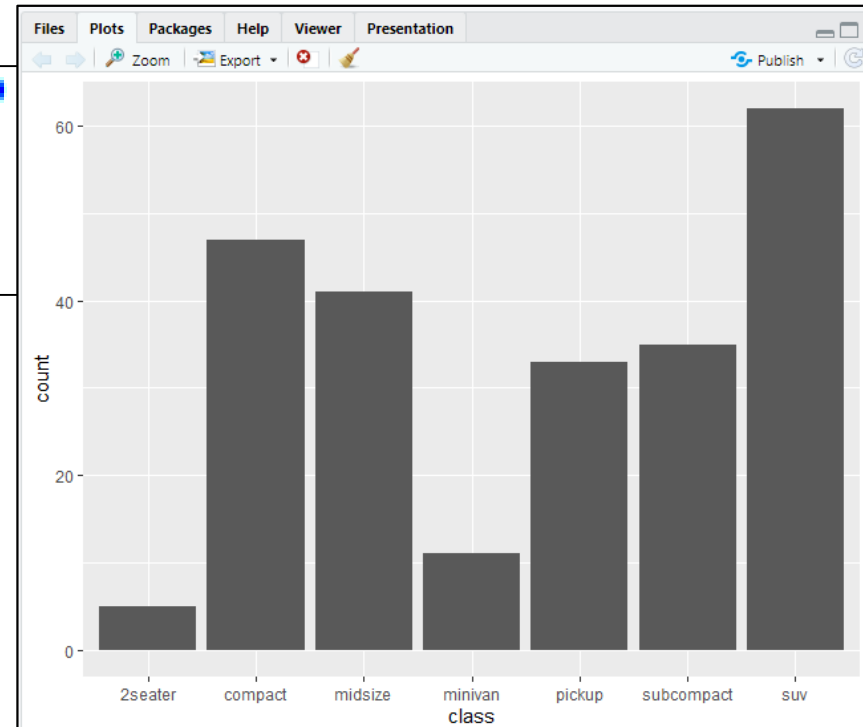


### 3. BARPLOTS

**Paso 4.** Haremos un barplot con una variable discreta que será class. Agruparemos los valores en función de sus palabras. Creamos un gráfico con datos y le decimos la columna del eje de las X. Le añadimos la función `geom_bar()` para crear un diagrama de barras con una variable discreta.

```
> grafico = ggplot(datos, aes(x=class))  
> grafico = grafico + geom_bar()  
> print(grafico)  
> |
```

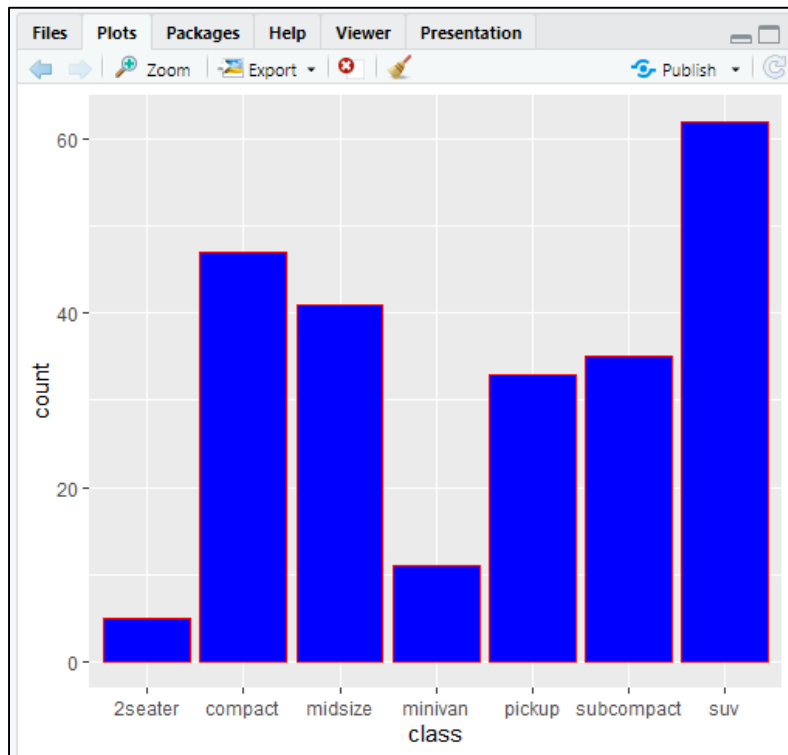
Ha creado un barplot con la variable discreta class con diferentes valores (Compact, Midsize, Minivan, etc) junto con el n° de veces que se repiten



### 3. BARPLOTS

**Paso 5.** Podemos cambiar el `geom_bar` y ponerle atributos, como por ejemplo el color: el contorno de color rojo y el relleno de color azul

```
> grafico = grafico + geom_bar(color='red',fill='blue')  
> print(grafico)  
> |
```

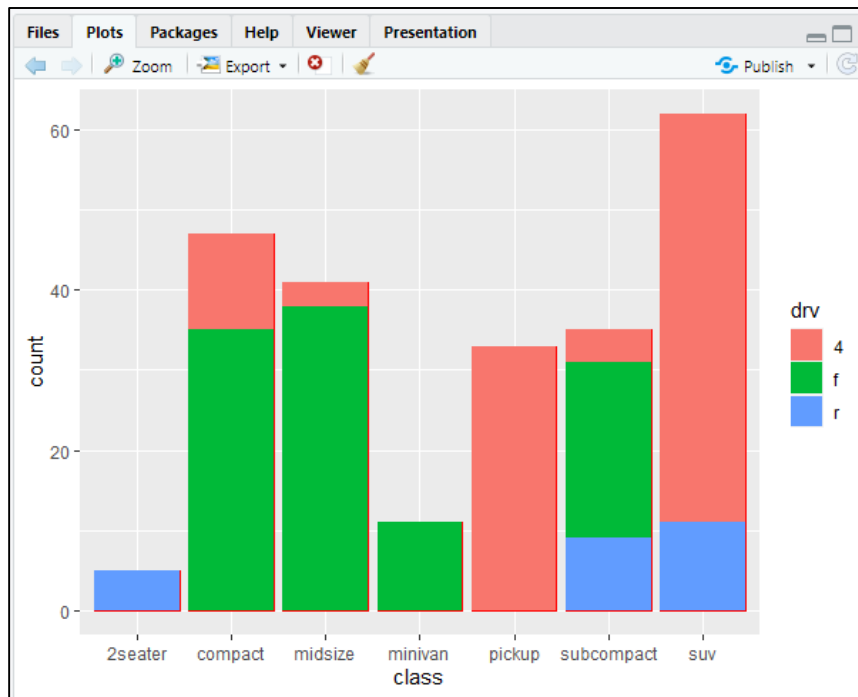


En lugar de color rojo,  
se puede poner su  
código alfanumérico

### 3. BARPLOTS

**Paso 6.** También podemos configurar en lugar de un color sólido, diferentes tipos de colores en función de una columna, por ejemplo, drv que tiene diferentes valores.

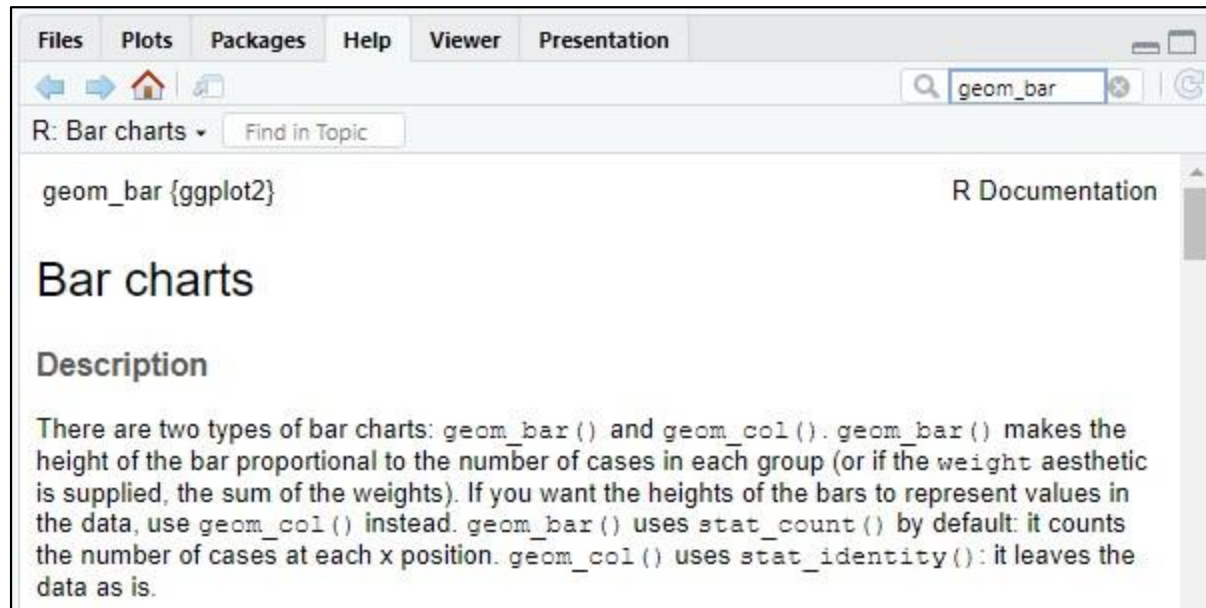
```
> grafico = grafico + geom_bar(aes(fill=drv))  
> print(grafico)  
>
```



En lugar de ser todo azul, ahora lo ha rellenado con los valores de drv. Podemos así comparar dos columnas, la columna class con su frecuencia y además sabiendo de qué tipo son por medio de otra columna drv

### 3. BARPLOTS

**Paso 7.** Podemos buscar información de `geom_bar` dentro de la pestaña help. Nos indica que es un diagrama de barras con la descripción de todos los argumentos que podemos ponerle. Al final suelen venir ejemplos de utilización del plot que estamos buscando.



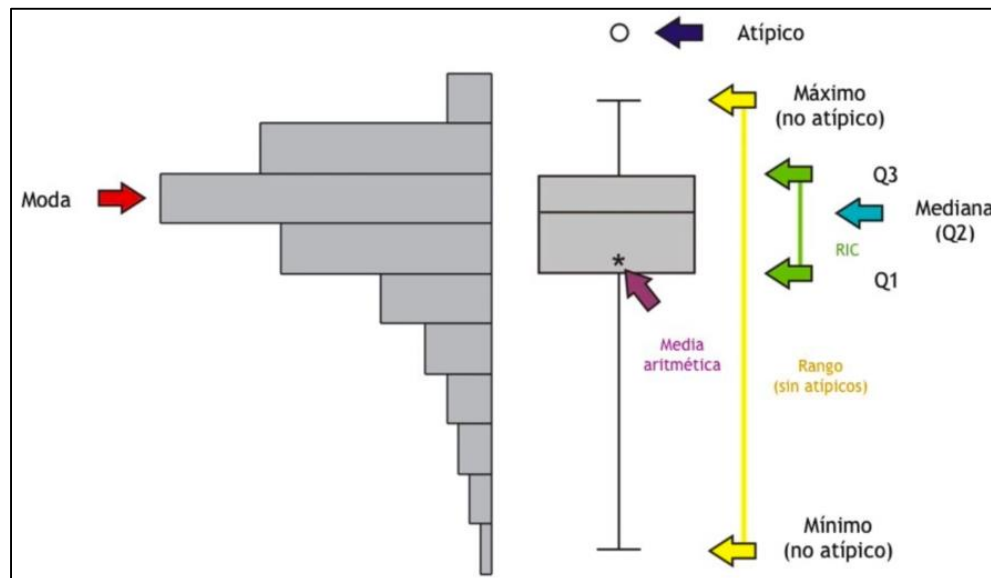
## 4. BOXPLOTS

---

- Los boxplots o diagramas de caja, es un método estandarizado para representar gráficamente una serie de datos numéricos a través de sus cuartiles
- Los diagramas de caja utilizan el cálculo de percentiles para determinar los valores de los cuartiles.
- Los cuartiles representan un método para dividir valores numéricos en 3 grupos donde el primer cuartil representa el 25%, el segundo el 50% y el tercero otro 25%.
- Muestra en una vista simple sus valores clave: la mediana, los cuartiles de los datos, y también representa sus valores atípicos (máximo y mínimo)

## 4. BOXPLOTS

- Vemos la representación de un grafico de barras habitual y su transformación a un gráfico de boxplot, donde se representa el 25% primero de los datos en la parte baja, otro 25% de los datos en la parte superior y la caja de en medio representa el 50% de los datos.
- El 50% de los datos están representados en la caja, donde la línea intermedia sería la mediana.



## 4. BOXPLOTS

**Paso 1.** Primero cargamos ggplot2 en memoria. Creamos una variable datos a partir del dataset mtcars.

Hacemos un head y vemos la información de coches en diferentes columnas

```
> library(ggplot2)
> datos = mtcars
> head(datos)
```

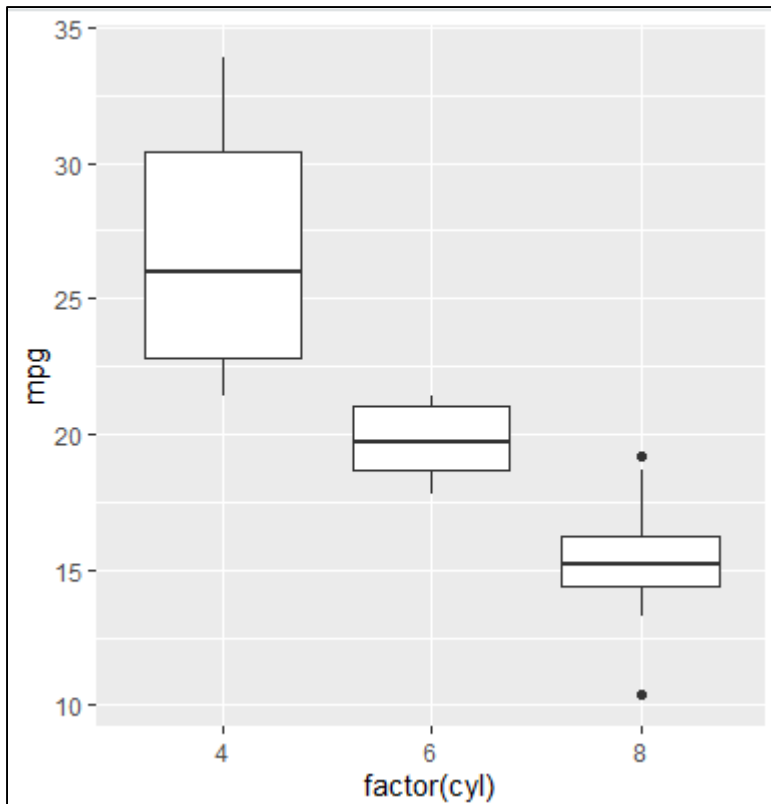
|                   | mpg  | cyl | disp | hp  | drat | wt    | qsec  | vs | am | gear | carb |
|-------------------|------|-----|------|-----|------|-------|-------|----|----|------|------|
| Mazda RX4         | 21.0 | 6   | 160  | 110 | 3.90 | 2.620 | 16.46 | 0  | 1  | 4    | 4    |
| Mazda RX4 wag     | 21.0 | 6   | 160  | 110 | 3.90 | 2.875 | 17.02 | 0  | 1  | 4    | 4    |
| Datsun 710        | 22.8 | 4   | 108  | 93  | 3.85 | 2.320 | 18.61 | 1  | 1  | 4    | 1    |
| Hornet 4 Drive    | 21.4 | 6   | 258  | 110 | 3.08 | 3.215 | 19.44 | 1  | 0  | 3    | 1    |
| Hornet Sportabout | 18.7 | 8   | 360  | 175 | 3.15 | 3.440 | 17.02 | 0  | 0  | 3    | 2    |
| Valiant           | 18.1 | 6   | 225  | 105 | 2.76 | 3.460 | 20.22 | 1  | 0  | 3    | 1    |

```
> |
```

## 4. BOXPLOTS

### Paso 2. Vamos a crear un grafico boxplot

```
> grafico = ggplot(datos, aes(x=factor(cyl),y=mpg))  
> grafico = grafico + geom_boxplot()  
> print(grafico)  
> |
```



El eje X es de tipo factor, donde se categoriza por los valores únicos de la columna cyl: 4, 6 y 8. El eje Y es mpg. Al añadir al gráfico la función `geom_boxplot` sin parámetros se crean 3 diagramas de caja para los valores de cyl 4, 6 y 8.

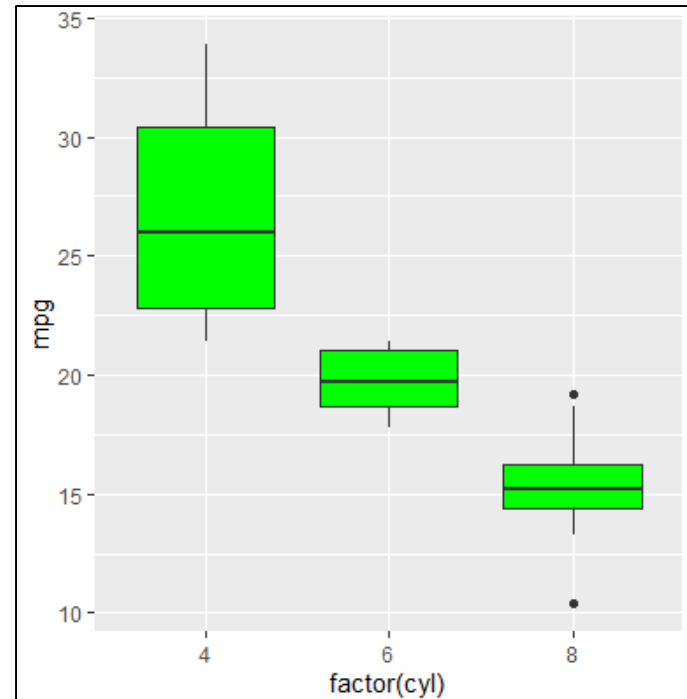
Para cyl=4 los valores de mpg van de 22 a 34. La caja representa al 50% de los valores que tiene mpg. La raya negra es la mediana 26



## 4. BOXPLOTS

**Paso 3.** Podemos cambiar los colores de relleno de las cajas. Sólo tenemos que configurar el `geom_boxplot`, y rellenamos el fondo las cajas de color verde, por ejemplo

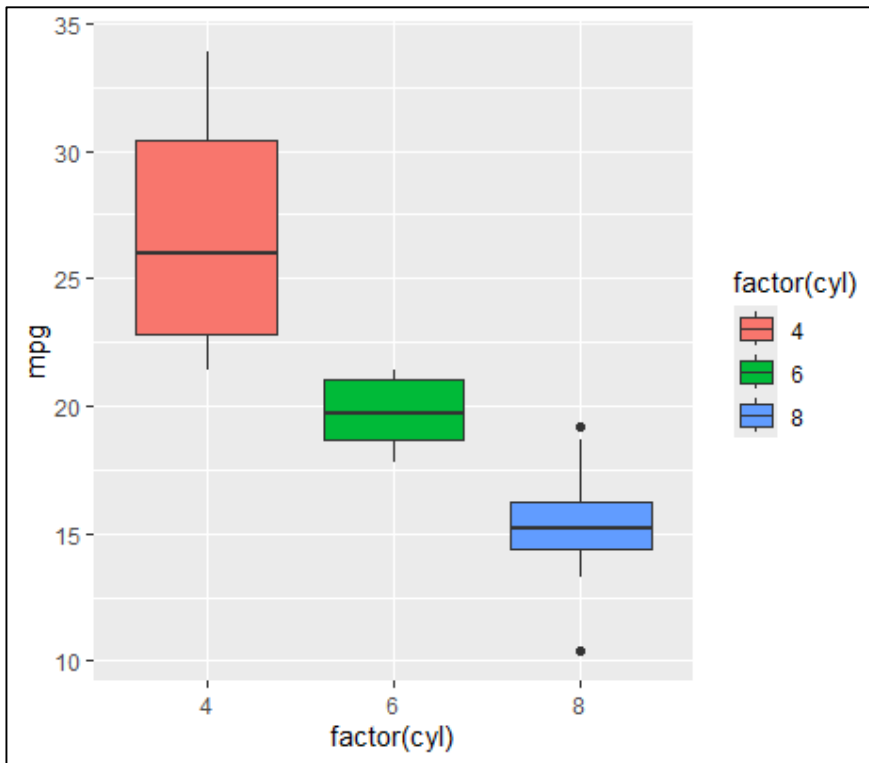
```
> grafico = grafico + geom_boxplot(fill='green')  
> print(grafico)  
>
```



## 4. BOXPLOTS

**Paso 4.** También podemos poner colores diferentes para los diagramas de caja en función del factor de cyl: 4, 6 y 8

```
> grafico = grafico + geom_boxplot(aes(fill=factor(cyl)))  
> print(grafico)  
> |
```



Han cambiado los colores en función del factor Cyl

Si cyl=4 es rojo

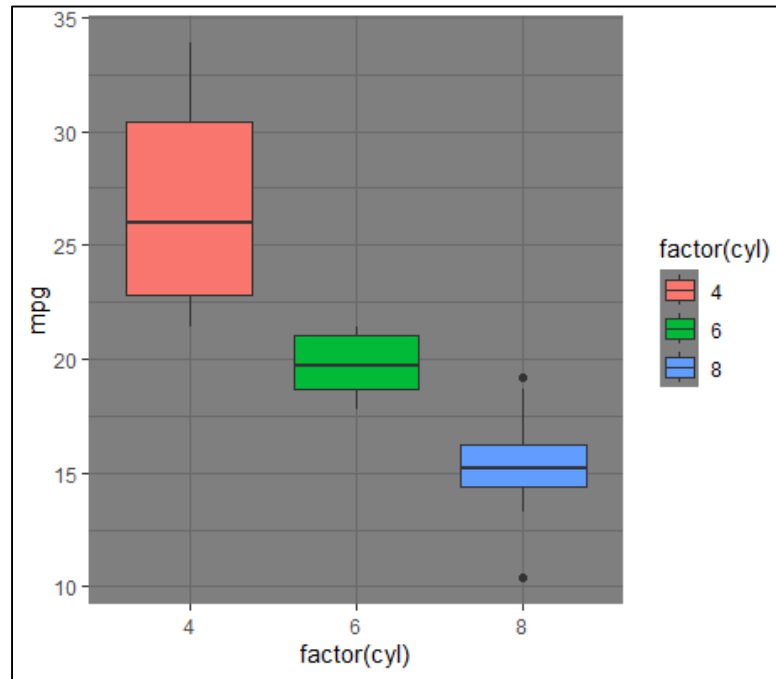
Si cyl=6 es verde

Si cyl=8 es azul

## 4. BOXPLOTS

**Paso 5.** También existen una serie de temas o estilos para el gráfico que se pueden utilizar. Por ejemplo si queremos que el fondo sea oscuro, podemos agregar al gráfico el tema dark.

```
> grafico = grafico + theme_dark()
> print(grafico)
> |
```

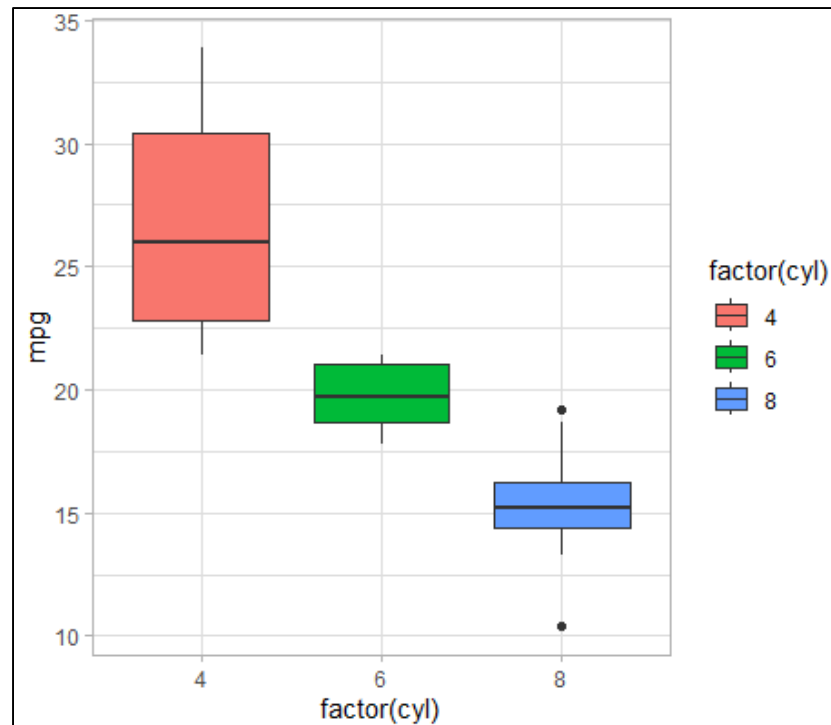


Hay muchas mas estilos que podemos utilizar para nuestro grafico

## 4. BOXPLOTS

**Paso 6.** Podemos agregar el tema light y volvemos a tener el mismo fondo claro de antes

```
> grafico = grafico + theme_light()
> print(grafico)
>
```



## 5. GRAFICOS PARA LA DISTRIBUCION DE 2 VARIABLES

---

**Paso 1.** Vamos a ver ahora en gráficos de distribución de dos variables. Para ello a parte de cargar la librería ggplot2, cargaremos el paquete ggplot2movies

```
install.packages('ggplot2movies')
```

```
library(ggplot2movies)
```

```
library(ggplot2)
```

## 5. GRAFICOS PARA LA DISTRIBUCION DE 2 VARIABLES

Paso 2. Dentro del paquete ggplot2movies viene el dataset movies, que asignaremos a una variable películas.

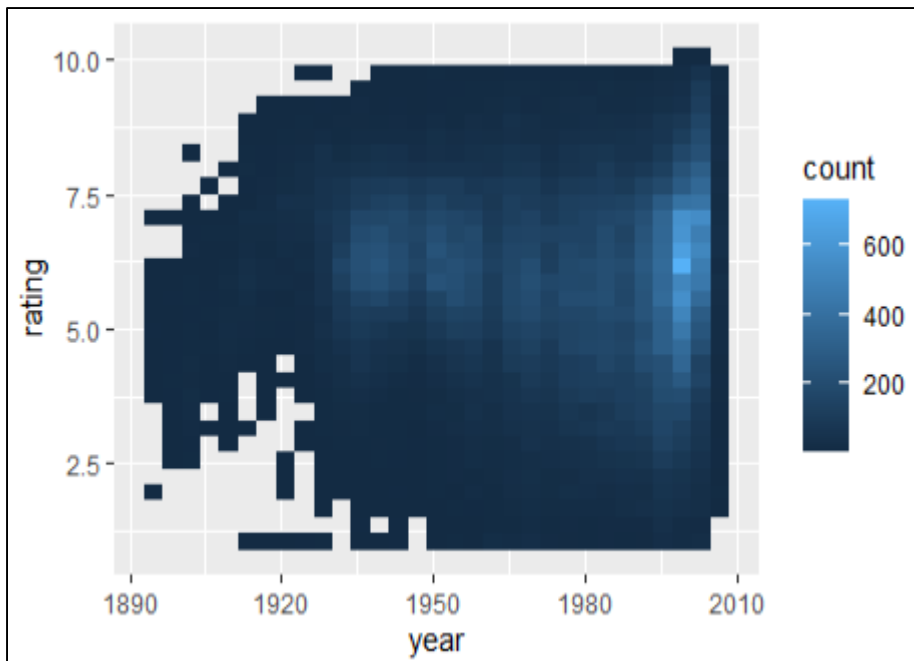
Si vemos las primeras líneas, es un dataset que contiene el titulo de la película, año de creación, ratings, votos, etc

```
> peliculas = movies
> head(peliculas)
# A tibble: 6 × 24
  title      year length budget rating votes  r1  r2  r3  r4
  <chr>    <int> <int> <int> <dbl> <int> <dbl> <dbl> <dbl> <dbl>
1 $      1971   121    NA    6.4   348  4.5  4.5  4.5  4.5
2 $1000 a T... 1939    71    NA    6     20  0   14.5  4.5  24.5
3 $21 a Day... 1941     7    NA    8.2    5  0    0    0    0
4 $40,000      1996    70    NA    8.2    6 14.5  0    0    0
5 $50,000 c... 1975    71    NA    3.4   17 24.5  4.5  0   14.5
6 $spent      2000    91    NA    4.3   45  4.5  4.5  4.5  14.5
# i 14 more variables: r5 <dbl>, r6 <dbl>, r7 <dbl>, r8 <dbl>,
#   r9 <dbl>, r10 <dbl>, mpaa <chr>, Action <int>, Animation <int>,
#   Comedy <int>, Drama <int>, Documentary <int>, Romance <int>,
#   Short <int>
> |
```

## 5. GRAFICOS PARA LA DISTRIBUCION DE 2 VARIABLES

**Paso 3.** Vamos a crear un gráfico para ver la concurrencia y relación entre dos variables. En el ejeX pondremos el año y en el ejeY el rating o puntuación que tiene esa película.

```
> grafico = ggplot(peliculas, aes(x=year, y=rating))  
> grafico = grafico + geom_bin2d()  
> print(grafico)  
>
```



Tenemos dos zonas mas claras donde el número de veces que se repite una valoración es alto.

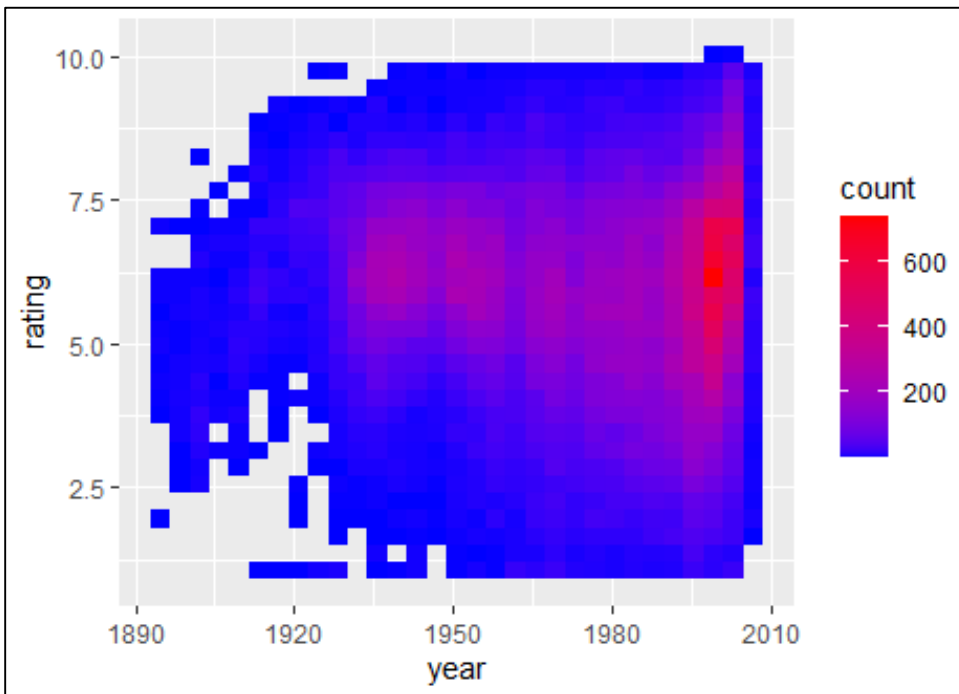
Alrededor del año 2000 hay muchas películas con un rating entre el 5 y el 7

Y alrededor de los años 50

## 5. GRAFICOS PARA LA DISTRIBUCION DE 2 VARIABLES

**Paso 4.** Si queremos cambiar los colores y poner un color gradiente, usaremos `scale_fill_gradient`, donde el color bajo será el azul y el color alto el rojo.

```
> grafico = grafico + scale_fill_gradient(low='blue', high='red')  
> print(grafico)  
> |
```



Con estos colores se identifican mejor las zonas con una mayor frecuencia de valores de rating por año.

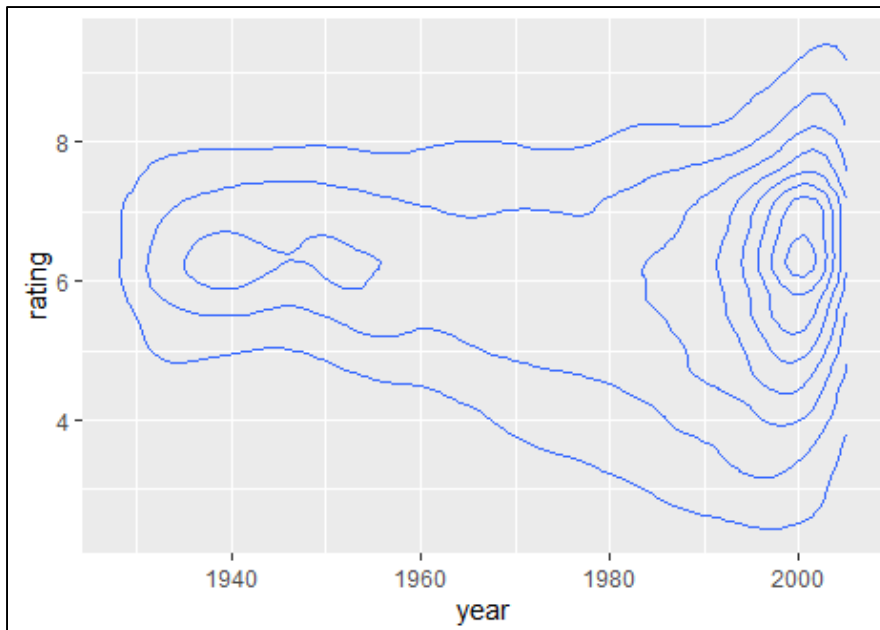
Las zonas con mayor frecuencia de rating, tiene un número de rating entre el 5 y el 7



## 5. GRAFICOS PARA LA DISTRIBUCION DE 2 VARIABLES

**Paso 5.** También podemos utilizar el tipo de gráfico `geom_density2d`. Hay que partir del gráfico inicial para que el otro no esté superpuesto. Y le agregamos la función `geom_density2d` sin ningún parámetro

```
> grafico = ggplot(peliculas, aes(x=year, y=rating))  
> grafico = grafico + geom_density2d()  
> print(grafico)  
> |
```



Otro tipo de gráfico, donde se concentran los círculos más pequeños, es donde hay un mayor número de cambios de frecuencia de pares de valores año y rating.

Hay una mayor concentración del 6.5, cerca del año 2000

## 6. LIMITES Y DIMENSIONES DE LOS GRAFICOS

**Paso 1.** Vamos a ver los límites y dimensiones de un gráfico. Cargamos la librería ggplot2, con la que crearemos un gráfico de puntos scatterplot para dos variables del dataset de coches mpg.

Creamos la variable datos con este dataset y observamos sus primeras filas: constructor de coches, modelo, año, etc

En la pestaña Environment vemos que la variable datos tiene 234 observaciones o filas por 11 variables o columnas.

```
> library(ggplot2)
> datos = mpg
> head(datos)
# A tibble: 6 × 11
  manufacturer model displ year   cyl trans      drv    cty   hwy fl      class
  <chr>         <chr> <dbl> <int> <int> <chr>    <chr> <int> <int> <chr>    <chr>
1 audi         a4      1.8  1999     4 auto(l5) f        18    29 p      compact
2 audi         a4      1.8  1999     4 manual(m5) f        21    29 p      compact
3 audi         a4      2    2008     4 manual(m6) f        20    31 p      compact
4 audi         a4      2    2008     4 auto(av)  f        21    30 p      compact
5 audi         a4      2.8  1999     6 auto(l5) f        16    26 p      compact
6 audi         a4      2.8  1999     6 manual(m5) f        18    26 p      compact
> |
```

## 6. LIMITES Y DIMENSIONES DE LOS GRAFICOS














---

**Paso 2.** Una vista mas clara la tenemos con la función str.  
Tiene 234 filas por 11 columnas

```
> str(datos)
tibble [234 × 11] (S3: tbl_df/tbl/data.frame)
 $ manufacturer: chr [1:234] "audi" "audi" "audi" "audi" ...
 $ model       : chr [1:234] "a4" "a4" "a4" "a4" ...
 $ displ      : num [1:234] 1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...
 $ year       : int [1:234] 1999 1999 2008 2008 1999 1999 2008 1999 1999 2008 ...
 $ cyl       : int [1:234] 4 4 4 4 6 6 6 4 4 4 ...
 $ trans      : chr [1:234] "auto(l5)" "manual(m5)" "manual(m6)" "auto(av)" ...
 $ drv       : chr [1:234] "f" "f" "f" "f" ...
 $ cty       : int [1:234] 18 21 20 21 16 18 18 18 16 20 ...
 $ hwy       : int [1:234] 29 29 31 30 26 26 27 26 25 28 ...
 $ fl       : chr [1:234] "p" "p" "p" "p" ...
 $ class     : chr [1:234] "compact" "compact" "compact" "compact" ...
> |
```

# 6. LIMITES Y DIMENSIONES DE LOS GRAFICOS

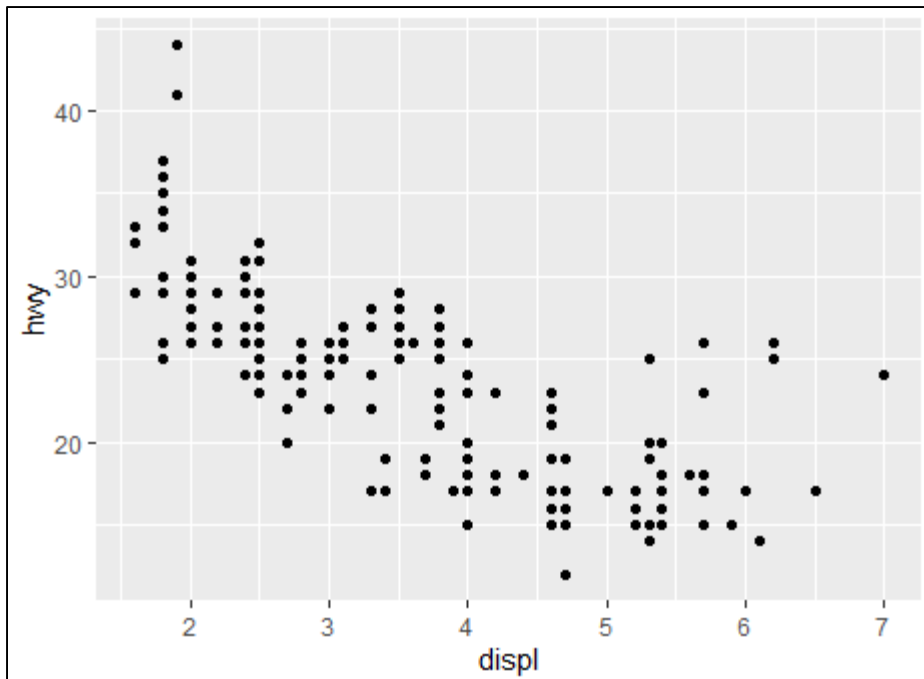
Paso 3. Creamos un grafico scatterplot, cuando hay dos variables o columnas con valores numéricos de forma continua. Utilizaremos la función `geom_point()`

| Dos Variables   |  |
|---|--|
| <p><b>X Continua, Y Continua</b><br/> <code>e &lt;- ggplot(mpg, aes(cty, hwy))</code></p> <p> <b>e + geom_label(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE)</b><br/> x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust</p> <p> <b>e + geom_jitter(height = 2, width = 2)</b><br/> x, y, alpha, color, fill, shape, size</p> <p> <b>e + geom_point()</b><br/> x, y, alpha, color, fill, shape, size, stroke</p> <p> <b>e + geom_quantile()</b><br/> x, y, alpha, color, group, linetype, size, weight</p> <p> <b>e + geom_rug(sides = "bl")</b><br/> x, y, alpha, color, linetype, size</p> <p> <b>e + geom_smooth(method = lm)</b><br/> x, y, alpha, color, fill, group, linetype, size, weight</p> <p> <b>e + geom_text(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE)</b><br/> x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust</p> | <p><b>Distribución Bivariada Continua</b><br/> <code>h &lt;- ggplot(diamonds, aes(carat, price))</code></p> <p> <b>h + geom_bin2d(binwidth = c(0.25, 500))</b><br/> x, y, alpha, color, fill, linetype, size, weight</p> <p> <b>h + geom_density2d()</b><br/> x, y, alpha, colour, group, linetype, size</p> <p> <b>h + geom_hex()</b><br/> x, y, alpha, colour, fill, size</p> <p><b>Función Continua</b><br/> <code>i &lt;- ggplot(economics, aes(date, unemploy))</code></p> <p> <b>i + geom_area()</b><br/> x, y, alpha, color, fill, linetype, size</p> <p> <b>i + geom_line()</b><br/> x, y, alpha, color, group, linetype, size</p> <p> <b>i + geom_step(direction = "hv")</b><br/> x, y, alpha, color, group, linetype, size</p> |

## 6. LIMITES Y DIMENSIONES DE LOS GRAFICOS

**Paso 4.** Creamos un grafico scatterplot donde el ejeX es la columna displ y el ejeY la columna hwy. De momento no le ponemos ningún tipo de parámetro al gráfico.

```
> grafico = ggplot(datos, aes(x=displ, y=hwy))  
> grafico = grafico + geom_point()  
> print(grafico)  
> |
```



Obtenemos el scatterplot de las dos variables del dataset mpg de los coches.

## 6. LIMITES Y DIMENSIONES DE LOS GRAFICOS

---

**Paso 5.** Si queremos recortar nuestro plot para visualizar mejor una zona, con la función `coord_cartesian` podemos establecer límites para los eje X e Y.

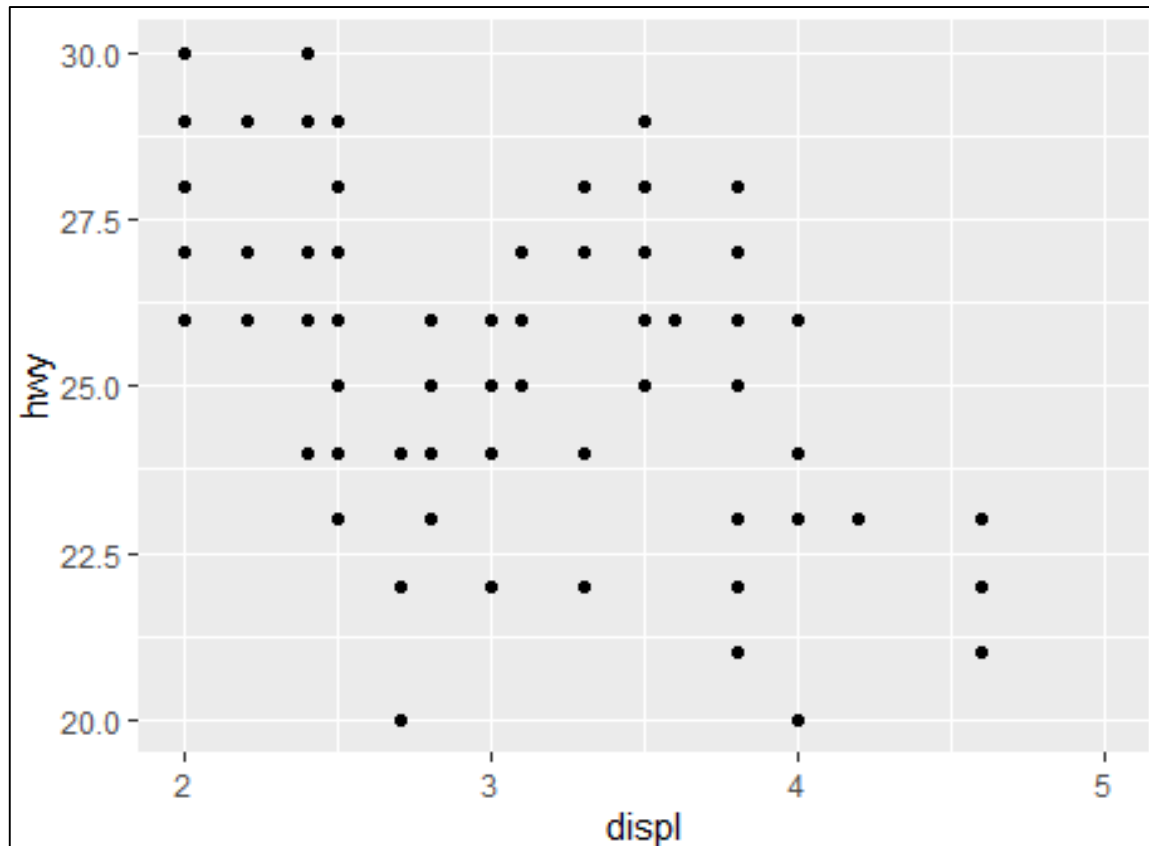
Nos permite focalizarnos en determinadas zonas o eliminar datos que no queremos que estén en el gráfico.

Con la función `coord_cartesian` estableceremos el limite para el ejeX entre 2 y 5, y para el eje Y entre 20 y 30.

```
> grafico = grafico + coord_cartesian(xlim=c(2,5),ylim=c(20,30))  
> print(grafico)  
> |
```

## 6. LIMITES Y DIMENSIONES DE LOS GRAFICOS

**Paso 6.** Obtenemos la focalización del grafico dentro de un nuevo rectángulo de limites  $[2,5] \times [20,30]$  para analizar mejor esa zona, cada punto, a modo de zoom

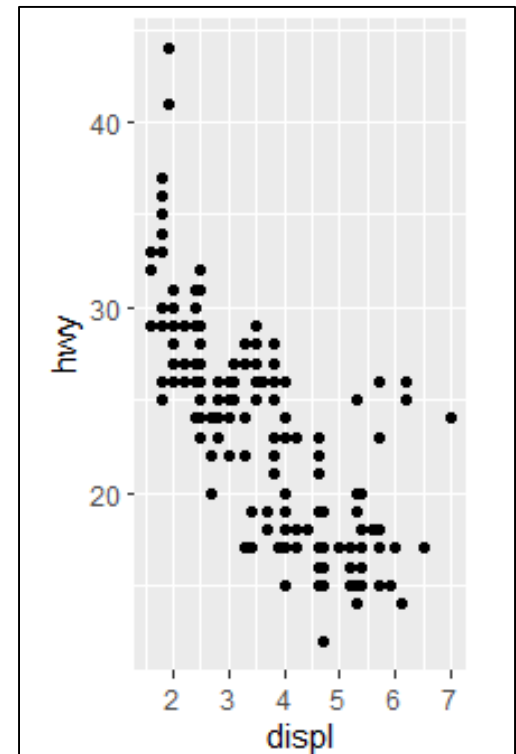


## 6. LIMITES Y DIMENSIONES DE LOS GRAFICOS

---

**Paso 7.** También podemos hacer un cambio a nivel de dimensiones. Volvemos al gráfico original y cambiamos sus dimensiones, mediante `coord_fixed`. Queremos que el eje Y tenga una dimensión que sea el triple del eje X. Si queremos que la X valga 1 y la Y 3, ponemos un ratio de 1/3. Por defecto el ratio entre X e Y es 1:1

```
> grafico = ggplot(datos, aes(x=displ, y=hwy))  
> grafico = grafico + geom_point()  
> print(grafico)  
> grafico = grafico + coord_fixed(ratio=1/3)  
> print(grafico)  
> |
```





## 7. GRAFICOS INTERACTIVOS CON PLOTLY

---

**Paso 1.** Plotly que es un paquete que sirve para crear gráficos interactivos. Instalamos el paquete y lo cargamos en memoria.

`install.packages('plotly')`

`library(plotly)`

Nuestra variable de datos vendrá del dataset mtcars. Vemos que las primeras líneas de datos es información de coches con sus características.

```
> library(ggplot2)
> datos = mtcars
> head(datos)
```

|                   | mpg  | cyl | disp | hp  | drat | wt    | qsec  | vs | am | gear | carb |
|-------------------|------|-----|------|-----|------|-------|-------|----|----|------|------|
| Mazda RX4         | 21.0 | 6   | 160  | 110 | 3.90 | 2.620 | 16.46 | 0  | 1  | 4    | 4    |
| Mazda RX4 wag     | 21.0 | 6   | 160  | 110 | 3.90 | 2.875 | 17.02 | 0  | 1  | 4    | 4    |
| Datsun 710        | 22.8 | 4   | 108  | 93  | 3.85 | 2.320 | 18.61 | 1  | 1  | 4    | 1    |
| Hornet 4 Drive    | 21.4 | 6   | 258  | 110 | 3.08 | 3.215 | 19.44 | 1  | 0  | 3    | 1    |
| Hornet Sportabout | 18.7 | 8   | 360  | 175 | 3.15 | 3.440 | 17.02 | 0  | 0  | 3    | 2    |
| valiant           | 18.1 | 6   | 225  | 105 | 2.76 | 3.460 | 20.22 | 1  | 0  | 3    | 1    |

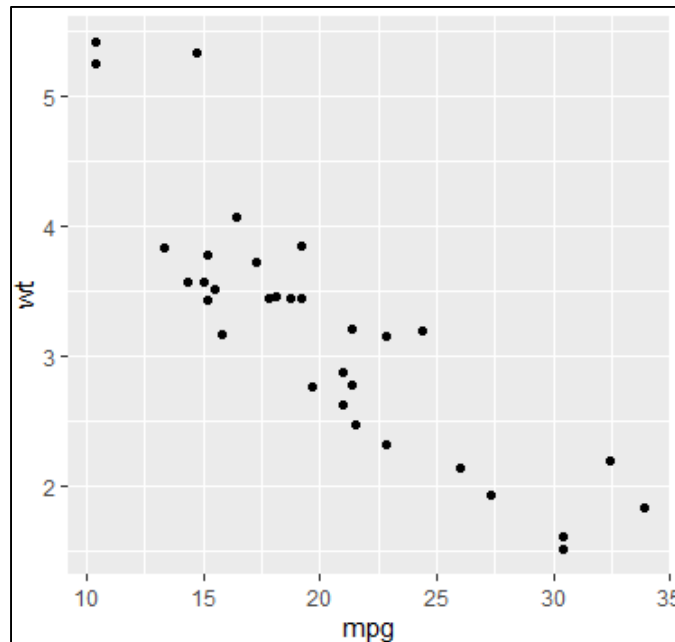
```
> |
```

## 7. GRAFICOS INTERACTIVOS CON PLOTLY

---

**Paso 2.** Hacemos un gráfico normal de tipo scatterplot. En el eje X ponemos la variable mpg y en el eje Y la variable wt. Es una simple grafica de comparación de dos variables, donde cada punto representa un par de valores MPG y WT.

```
> grafico = ggplot(datos, aes(mpg, wt)) + geom_point()  
> print(grafico)  
> |
```



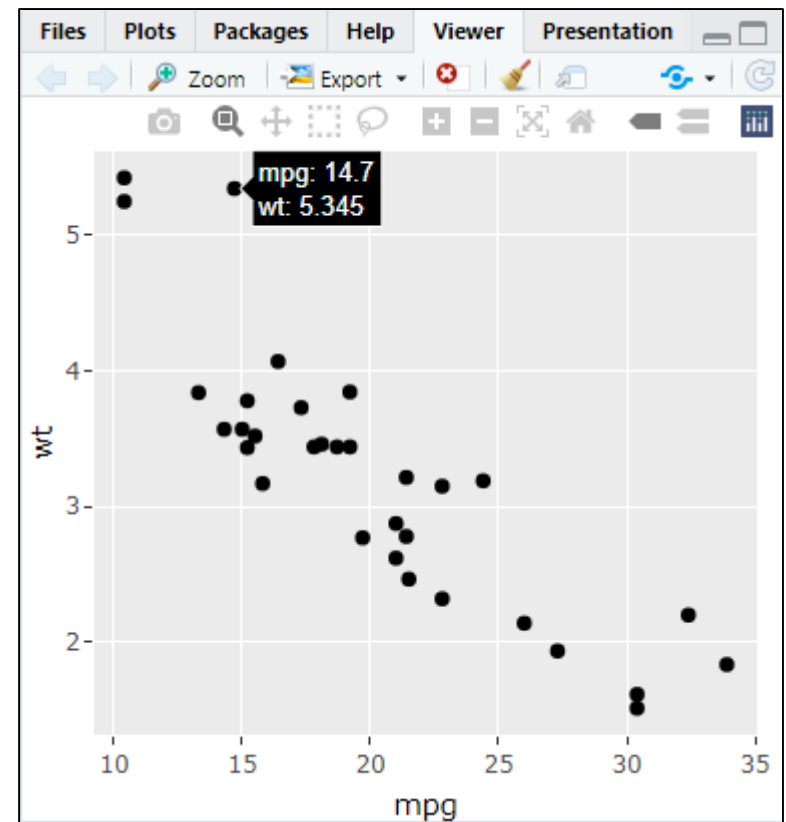
## 7. GRAFICOS INTERACTIVOS CON PLOTLY

**Paso 3.** Mediante plotly podemos convertir este gráfico en un gráfico interactivo. Creamos un segundo gráfico con la función `ggplotly`, y le pasamos como parámetro el gráfico que acabamos de crear.

```
> grafico2 = ggplotly(grafico)
> print(grafico2)
> |
```

Se ha convertido en un gráfico interactivo.

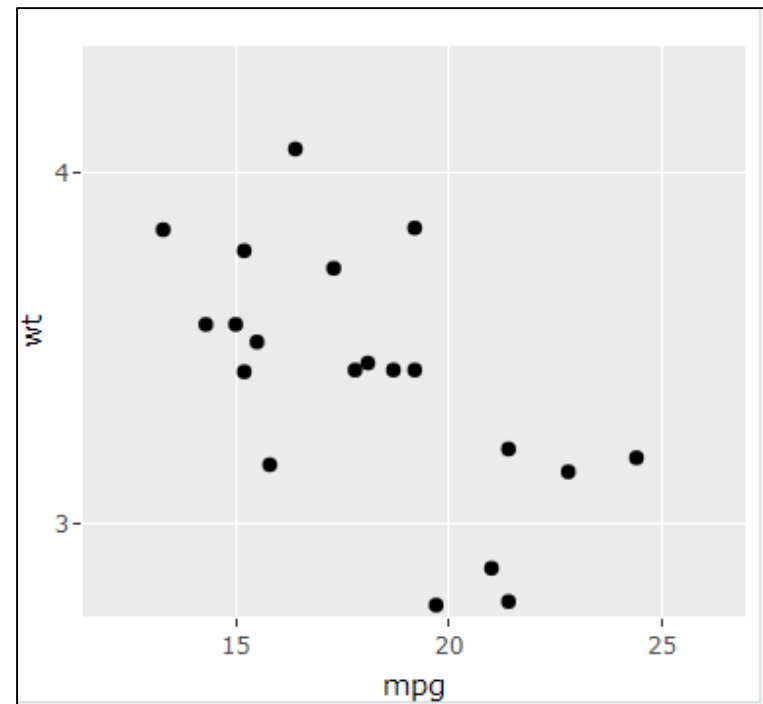
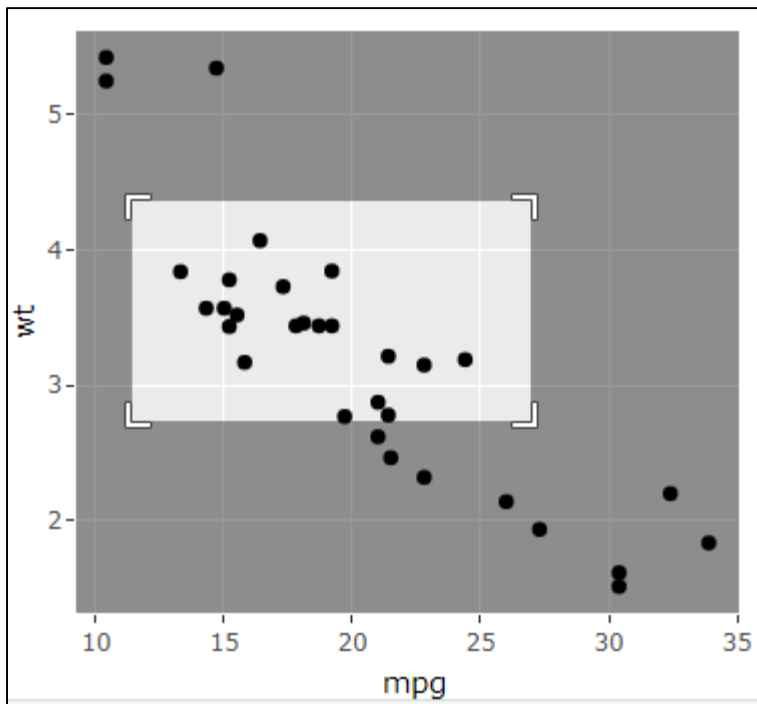
Si pulsamos encima de cada punto obtenemos los valores x (mpg) e y (wt) del punto, lo cual resulta muy útil



## 7. GRAFICOS INTERACTIVOS CON PLOTLY

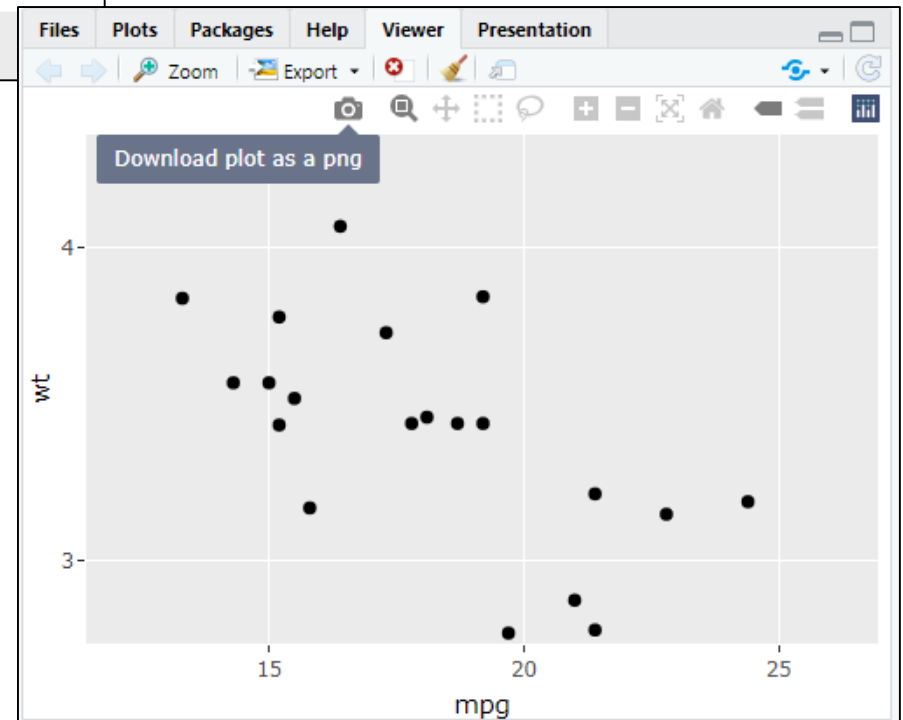
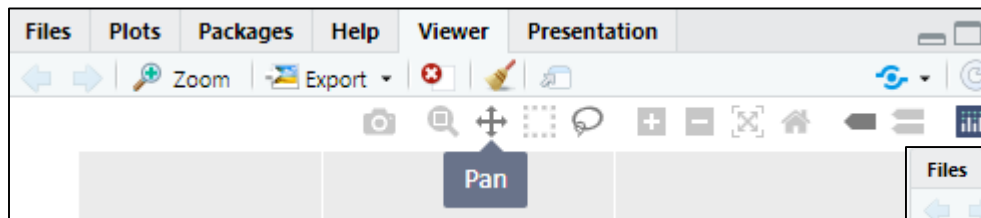
---

**Paso 4.** Este tipo de grafico, permite seleccionar ciertas zonas del grafico con el ratón, ampliandolas en pantalla. Permite ver de manera más clara la relación entre ellos. Si hacemos doble clic volvemos nuevamente a donde estábamos antes.



# 7. GRAFICOS INTERACTIVOS CON PLOTLY

**Paso 5.** Tiene diferentes funciones. Permite ampliar mas el grafico, desplazarse por el grafico, sacar una foto en formato png, etc.



# 7. GRAFICOS INTERACTIVOS CON PLOTLY

**Paso 6.** Podemos ir a la página <https://plotly.com/ggplot2/> donde podemos encontrar una gran cantidad de tipos de gráficos interactivos que se pueden utilizar con el paquete plotly desde R: fundamentales, básicos, estadísticos, científicos, etc

The screenshot shows the Plotly ggplot2 Open Source Graphing Library website. The page features a dark sidebar on the left with navigation links for 'Quick Reference' (Getting Started, Is Plotly Free?, GitHub, community.plotly.com) and 'Examples' (Fundamentals, Basic Charts). The main content area has a header with the Plotly logo and 'Graphing Libraries' text. Below this is a search bar and a section titled 'Plotly ggplot2 Open Source Graphing Library'. The text describes how to use ggplotly() to convert ggplot2 figures into interactive ones. It also mentions that ggplotly is free and open source, and provides links to view the source, report issues, or contribute on GitHub. A 'Deploy Ggplot2 AI Dash apps on private Kubernetes clusters' section includes links for Pricing, Demo, Overview, and AI App Services. At the bottom, there is a 'Fundamentals' section with a grid of various interactive charts, including a scatter plot, a line chart, a bar chart, and a stacked bar chart.

## 7. GRAFICOS INTERACTIVOS CON PLOTLY

---

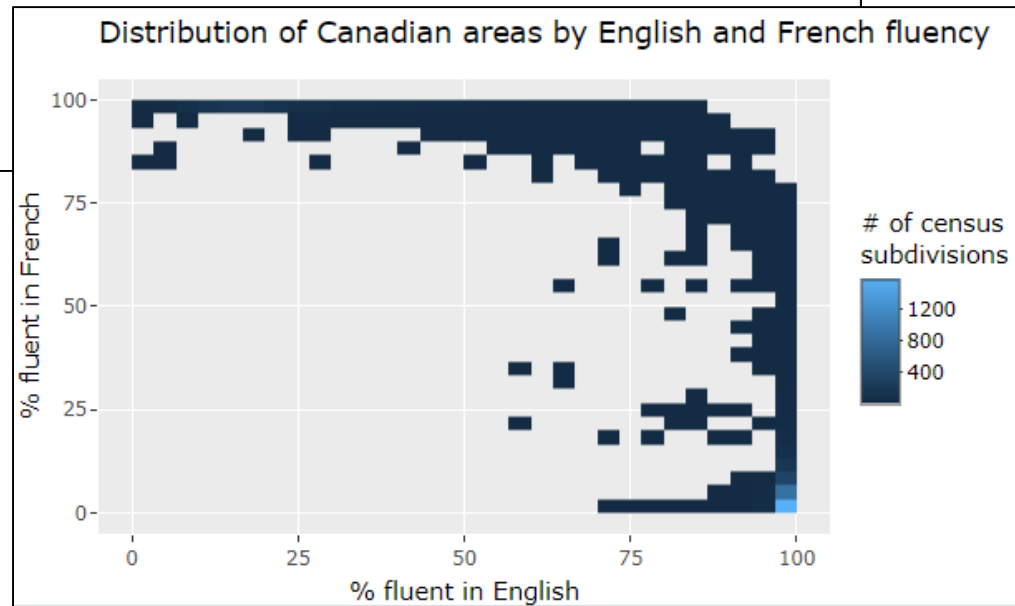
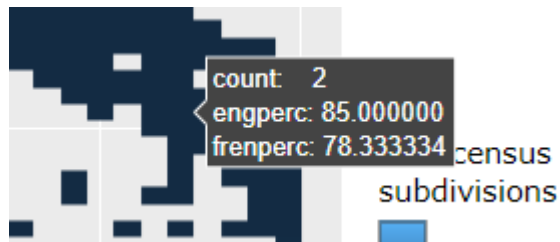
**Paso 7.** Si queremos utilizar alguno de ellos, sólo entramos, por ejemplo, en Headmap. Y aquí tenemos el código en R que podemos utilizar para construir un gráfico de este tipo

```
1 library(plotly)
2
3 english_french <- read.csv("https://raw.githubusercontent.com/plotly/datasets/master/english_french.csv", stringsAsFactors = FALSE)
4
5 p <- ggplot(english_french, aes(x=engperc,y=frenperc)) +
6   geom_bin2d() +
7   labs(title = "Distribution of Canadian areas by English and French fluency",
8        x = "% fluent in English",
9        y = "% fluent in French",
10        fill = "# of census \nsubdivisions")
11 ggplotly(p)
```

# 7. GRAFICOS INTERACTIVOS CON PLOTLY

**Paso 8.** Copiamos este código, lo pegamos en Rstudio, y obtenemos el mismo gráfico interactivo que podemos utilizar.

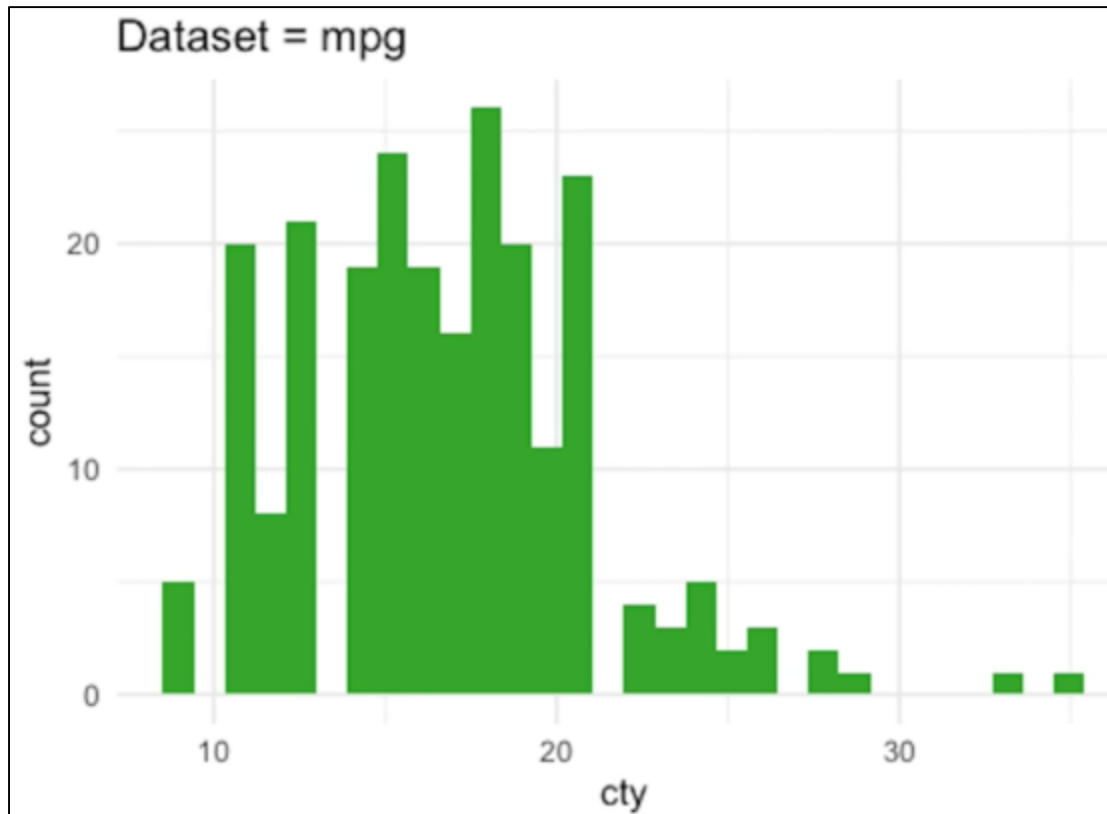
```
> library(plotly)
>
> english_french <- read.csv("https://raw.githubusercontent.com/plotly/datasets/master/english_french.csv", stringsAsFactors = FALSE)
>
> p <- ggplot(english_french, aes(x=engperc, y=frenperc)) +
+   geom_bin2d() +
+   labs(title = "Distribution of Canadian areas by English and French fluency",
+         x = "% fluent in English",
+         y = "% fluent in French",
+         fill = "# of census \nsubdivisions")
> ggplotly(p)
> |
```



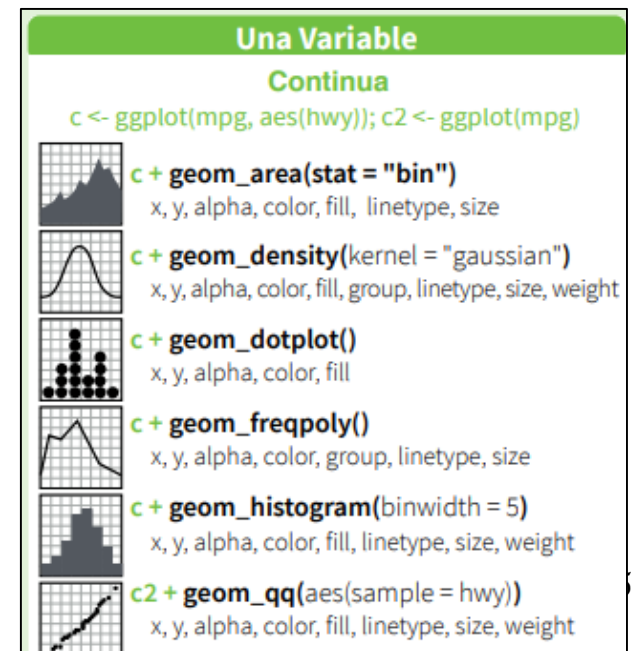


## 8. EJERCICIOS

**Ejercicio 1.** Reproducir este gráfico de tipo histograma, que proviene del dataset MPG. En el ejeX tenemos la variable *cty*, y en el ejeY el numero de veces que se repiten los valores de *cty*.

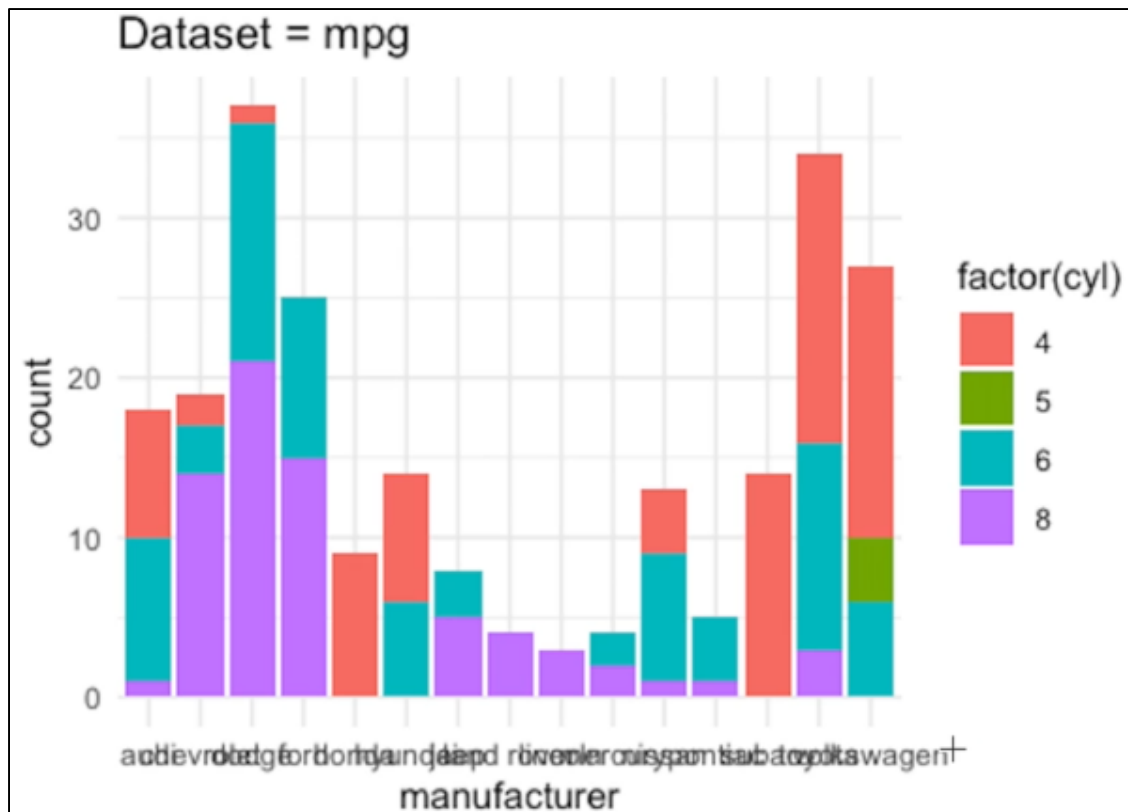


Se trata de un gráfico de una única variable y con valores continuos



## 8. EJERCICIOS

**Ejercicio 2.** Reproducir a partir del dataset mpg, el siguiente gráfico. Los valores de constructores están categorizados según el valores de cyl, y mostramos la frecuencia con la que se repiten los valores constructores



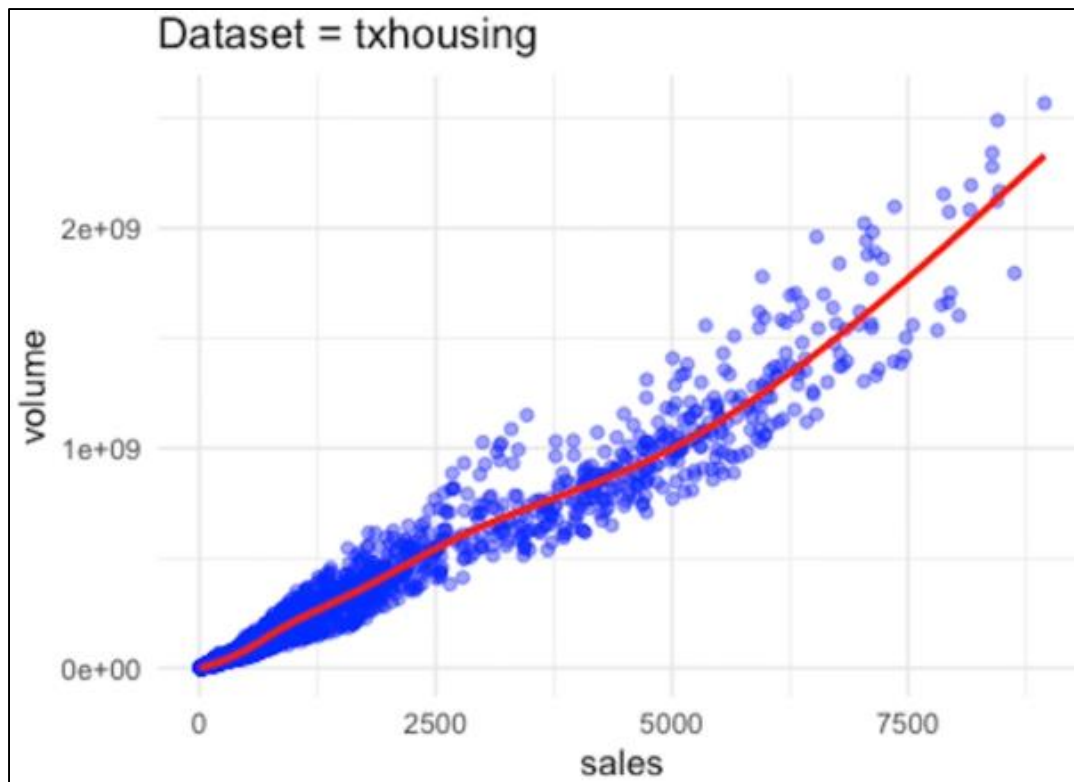
Se trata de un gráfico de una única variable manufacturer de tipo discreta

**Discreta**

```
d <- ggplot(mpg, aes(fl))  
d + geom_bar()  
x, alpha, color, fill, linetype, size, weight
```

## 8. EJERCICIOS

**Ejercicio 3.** Reproducir el siguiente grafico a partir del dataset txhousing, donde se comparan dos variables Sales y Volume, que son numéricas y continuas.



**X Continua, Y Continua**

```
e <- ggplot(mpg, aes(cty, hwy))
```



```
e + geom_label(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE)
```

x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust



```
e + geom_jitter(height = 2, width = 2)
```

x, y, alpha, color, fill, shape, size



```
e + geom_point()
```

x, y, alpha, color, fill, shape, size, stroke



```
e + geom_quantile()
```

x, y, alpha, color, group, linetype, size, weight



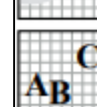
```
e + geom_rug(sides = "bl")
```

x, y, alpha, color, linetype, size



```
e + geom_smooth(method = lm)
```

x, y, alpha, color, fill, group, linetype, size, weight



```
e + geom_text(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE)
```

x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust