Contents

Chapter 1

Resumen

Este bookdown fue diseñado para ser presentado en la cuarta clase del ramo *Ciencias Sociales Computacionales: Análisis de Redes Sociales y Procesamiento de Lenguaje Natural*, impartida por el profesor Naim Bro Khomasi el segundo semestre del 2021.

Curso: Ciencias Sociales Computacionales - *Análisis de Redes Sociales y Procesamiento de Lenguaje Natural*.
Institución: Universidad de Chile.
Profesor: Naim Bro Khomasi.
Ayudantes: Jan Dimter y Nicolas Godoy.
Fecha: 26 de agosto de 2021.

Chapter 2

Introducción clase

En esta clase realizaremos ejercicios de visualización a través de las librerías **ggplot2** y **plotly**. Utilizaremos el dataset `mpg` para la creación de gráficos. Éste se encuentra contenido en *tidyverse* [1].

Realizamos la instalación y carga de los paquetes con el siguiente código:

```
#install.packages("tidyverse")
#install.packages("plotly")
library(tidyverse)
library(plotly)
```

2.1 Sobre el dataset

Según la documentación de mpg (puedes encontrarla usando en tu consola el comando `?mpg` una vez hayas instalado ggplot2), las variables son las siguientes:

Variable	Detalle
manufacturer	Marca del auto
model	Modelo del auto
displ	Cilindrada, en litros.
year	Año de manufactura
cyl	Número de cilindros
trans	Tipo de transmisión
drv	Tipo de transmisión (f = transmisión delantera, r = transmisión trasera, 4 = 4x4)

[1] Esta librería llama paquetes adicionales del universo tidyverse tales como ggplot2, dplyr, readr, stringr, etc.

Variable	Detalle
cty	Millas por galón en ciudad
hwy	Millas por galón en autopista
fl	Tipo de combustible
class	Tipo de auto

Una vez cargados los paquetes, revisamos las primeras y últimas filas de nuestro dataset:

```
# Usamos knitr::kable() para ajustar el resultado a la visualización html.
# Una alternativa más simple es usar por separadop las funciones head() y tail().
knitr::kable(mpg[c(1:3,(nrow(mpg)-2):nrow(mpg)),])
```

manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	class
audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
volkswagen	passat	2.8	1999	6	auto(l5)	f	16	26	p	midsize
volkswagen	passat	2.8	1999	6	manual(m5)	f	18	26	p	midsize
volkswagen	passat	3.6	2008	6	auto(s6)	f	17	26	p	midsize

Chapter 3

ggplot2: Componentes base

La librería ggplot2 de Hadley Wickham et al. (2014), parte del ecosistema de paquetes ***tidyverse,*** es un sistema declarativo para la creación de gráficos. ¿Qué quiere decir eso? Que le señalamos a ggplot2 cuál es nuestra data, cómo mapearla, qué elementos gráficos usar, y ggplot se encarga del resto.

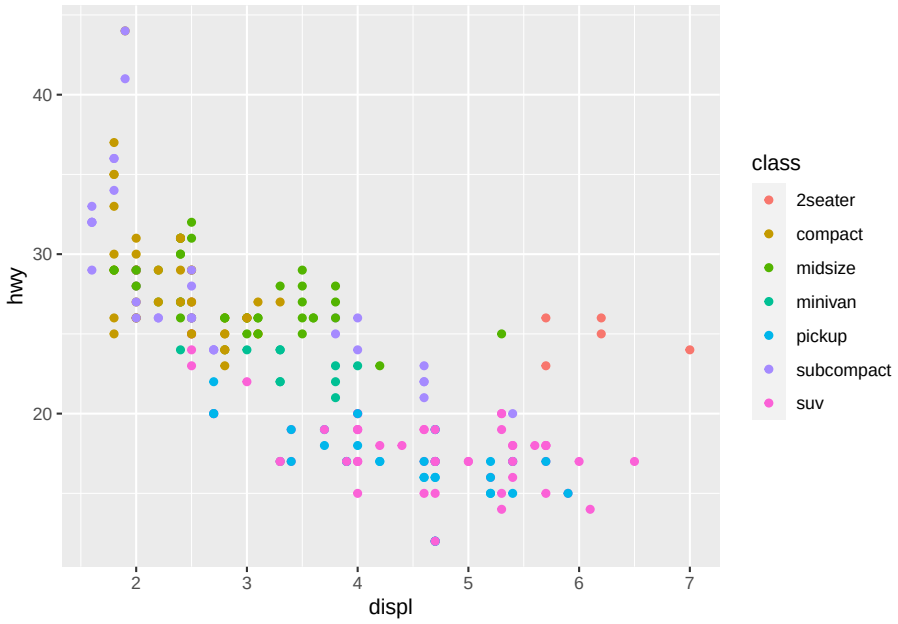
```
ggplot(mpg, aes(displ, hwy, colour = class)) +
  geom_point()
```

Son 3 los componentes base a la hora de trabajar con ggplot2:

1. **Data**: literalmente son los datos que le entregaremos. Comunmente serán *data frames* o similares. Pueden estar agrupados o no. En el caso de nuestro ejemplo corresponde a `mpg`.
2. **Aesthetic mapping:** corresponde a indicadores en torno a la data que usamos. Posiblemente los más comunes e importantes son los parámetros x e y (ej. en un diagrama de dispersión). También podemos incorporar distinción por colores, formas, etc. En nuestra sintaxis de ejemplo, corresponde a `aes(displ, hwy, colour = class)`.
3. **Geometry:** Hasta ahora hemos señalado nuestra data y qué representación le daremos en el gráfico, sin embargo, nos falta el último componente: ¿qué tipo de gráfico usaremos? Es posible generar puntos, barras, cajas, etc. Las geometrías suelen ser combinables dependiendo del

tipo de información con la que trabajes. En nuestro código de ejemplo corresponde a `geom_point()`.

El resultado de ejemplo es el siguiente:



Chapter 4

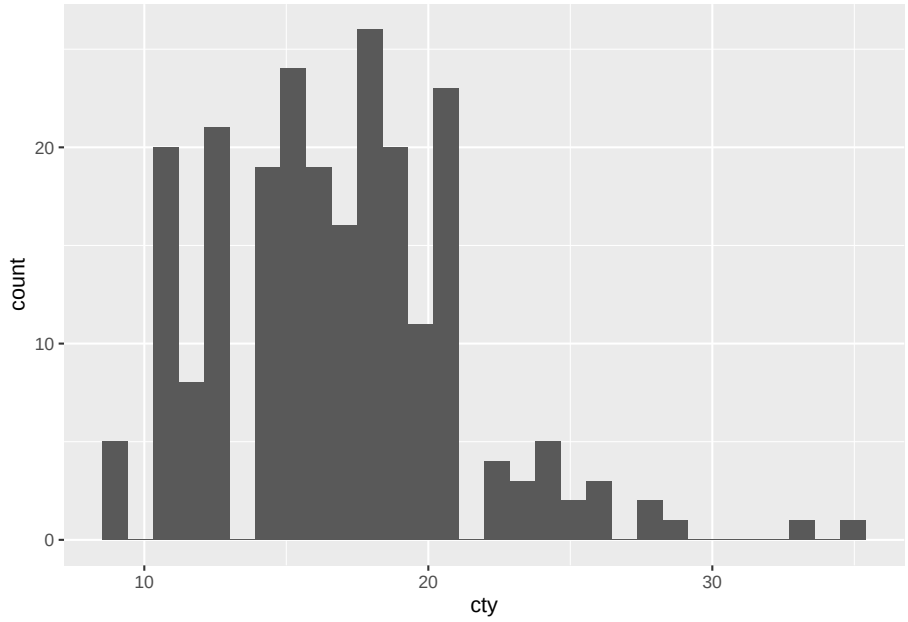
ggplot: Principales tipos de gráficos

Revisaremos rápidamente gráficos que puedes constuir usando ggplot. Usaremos para este fin mpg.

1. **Histograma** (histogram)

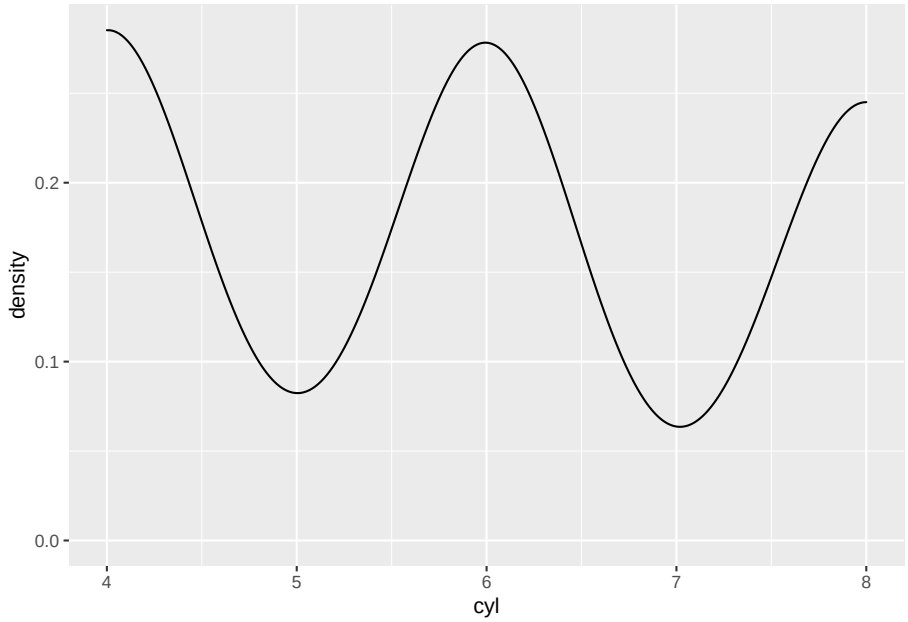
```
histograma <- ggplot(mpg, aes(x=cty)) +
  geom_histogram()
histograma
```

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



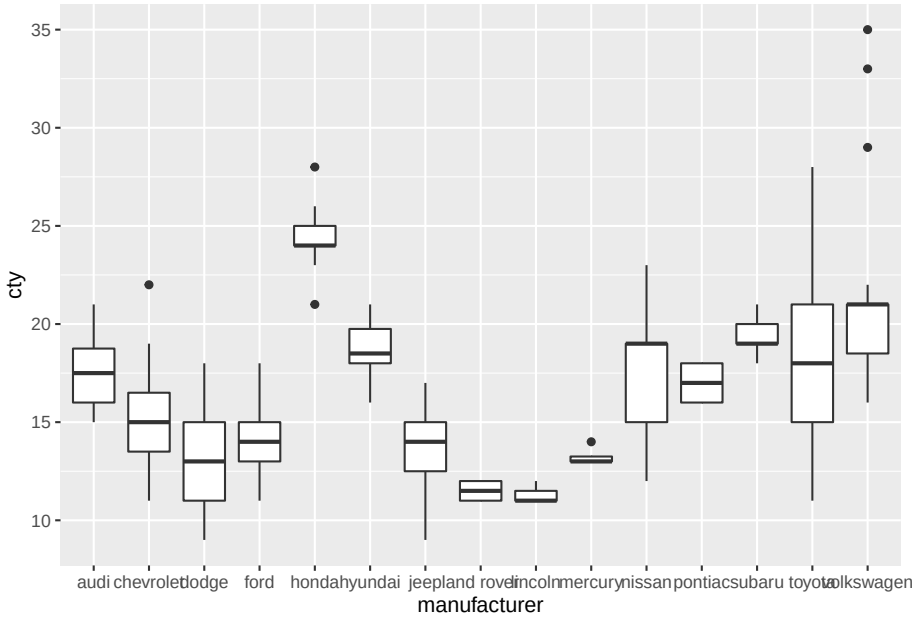
2. **Gráfico de densidad** (density plot)

```
densidad<-ggplot(mpg, aes(x=cyl))+
  geom_density()
densidad
```



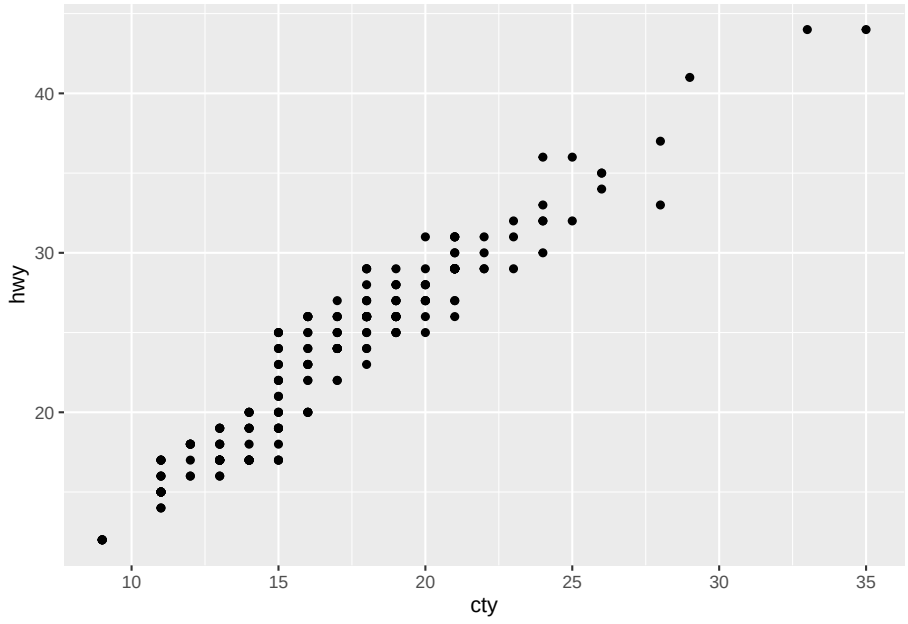
3. **Diagrama de cajas** (boxplot)

```
cajas<-ggplot(mpg, aes(x=manufacturer, y=cty)) +
    geom_boxplot()
cajas
```



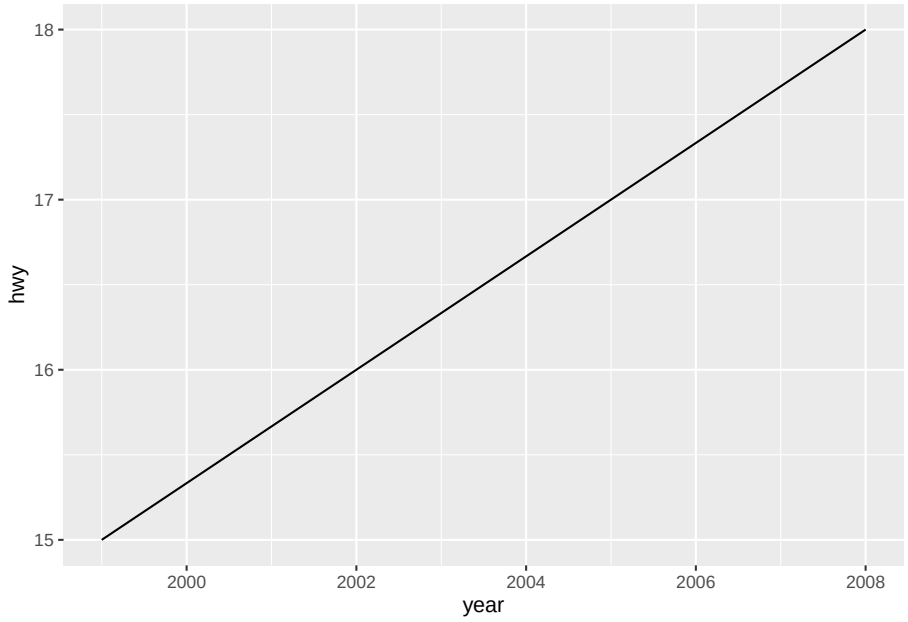
4. **Diagrama de dispersión** (scatterplot)

```
dispersion<-ggplot(mpg, aes(x=cty, y=hwy)) +
    geom_point()
dispersion
```



5. **Gráfico de líneas** (lineplot)

```
#Filtramos por un modelo en específico
lineas<-ggplot(mpg[mpg$model=="range rover",], aes(x=year, y=hwy, group=manufacturer))
  geom_line()
lineas
```



4.1 Intentemos mejorar la visualización

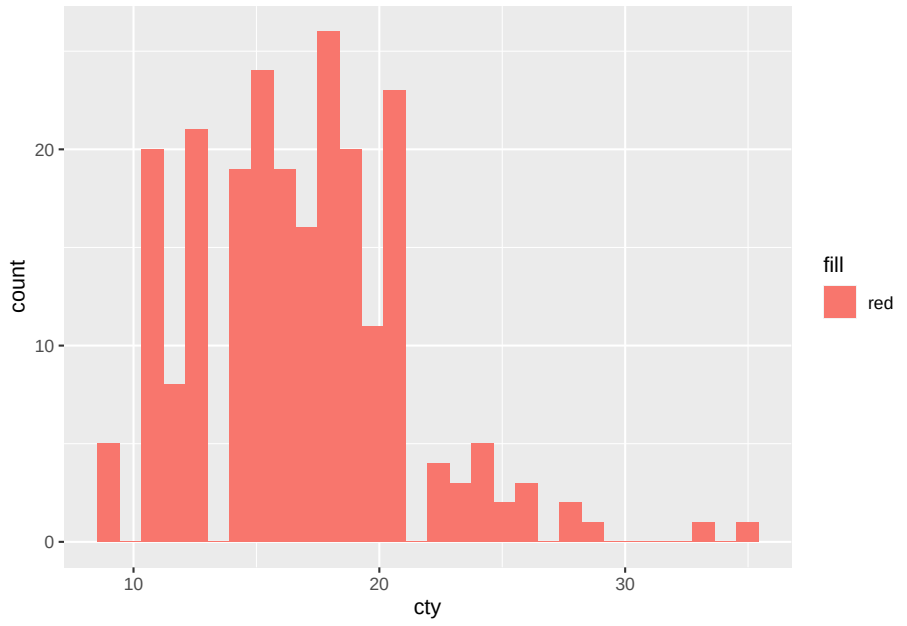
Si bien los ejemplos anteriores son simples en cuanto a código, la calidad de la visualización es poca. Podemos mejorar los gráficos utilizando argumentos y funciones.

4.1.1 Color

Una gran opción para mejorar nuestros gráficos es incorporar colores. Estos pueden ser iguales para todas las formas como aquí:

```
histograma <- ggplot(mpg, aes(x=cty, fill="red")) +
  geom_histogram()
histograma
```

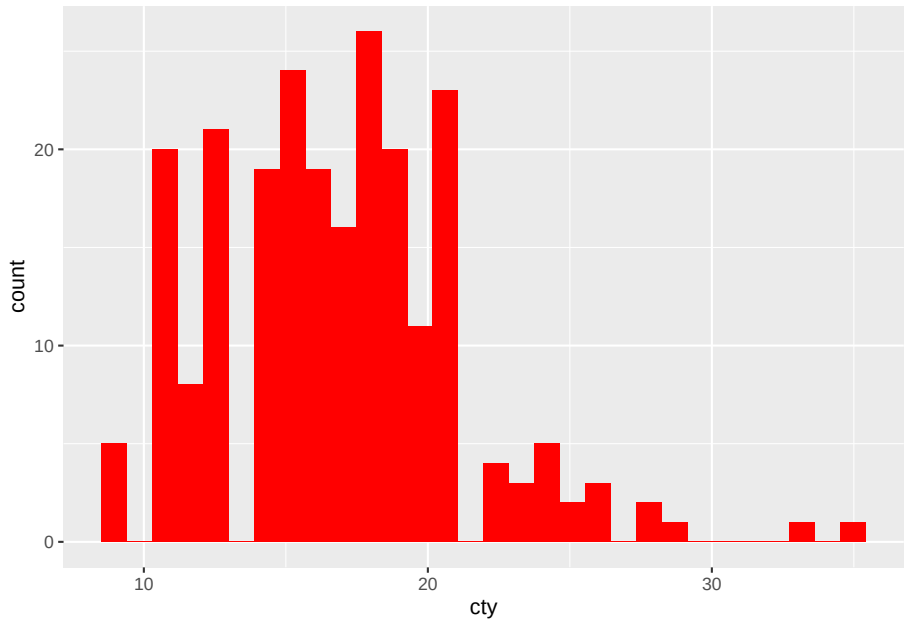
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



¿Lo ingresamos correctamente? ¿Es el color de relleno una variable categórica? Lo correcto es incorporarlo como característica de la forma geométrica:

```
histograma <- ggplot(mpg, aes(x=cty)) +
  geom_histogram(fill="red")
histograma
```

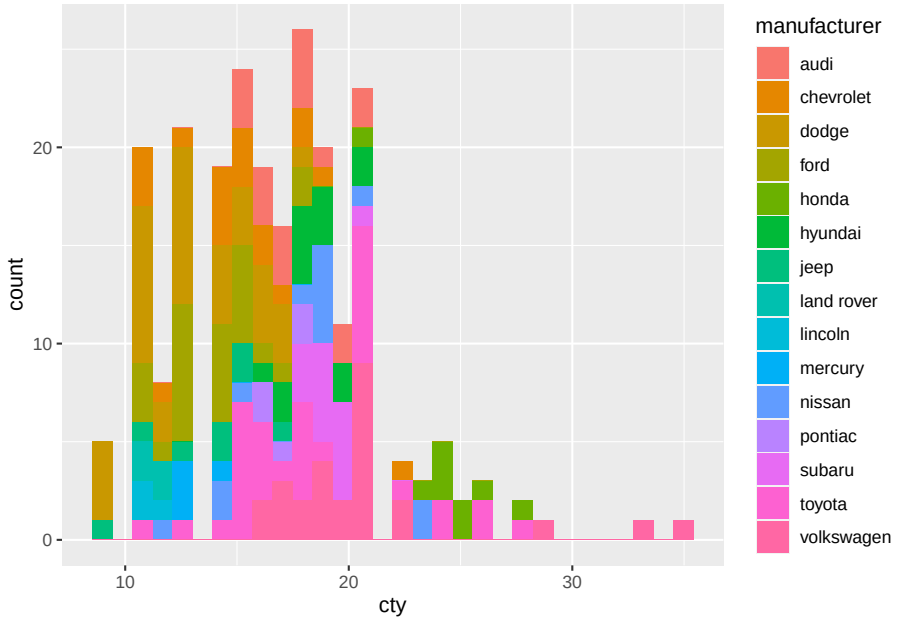
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



¿Y si queremos usar los colores para distinguir en relación a una variable categórica? Ahora sí lo incorporamos al mapeo estético:

```
histograma <- ggplot(mpg, aes(x=cty, fill=manufacturer)) +
  geom_histogram()
histograma
```

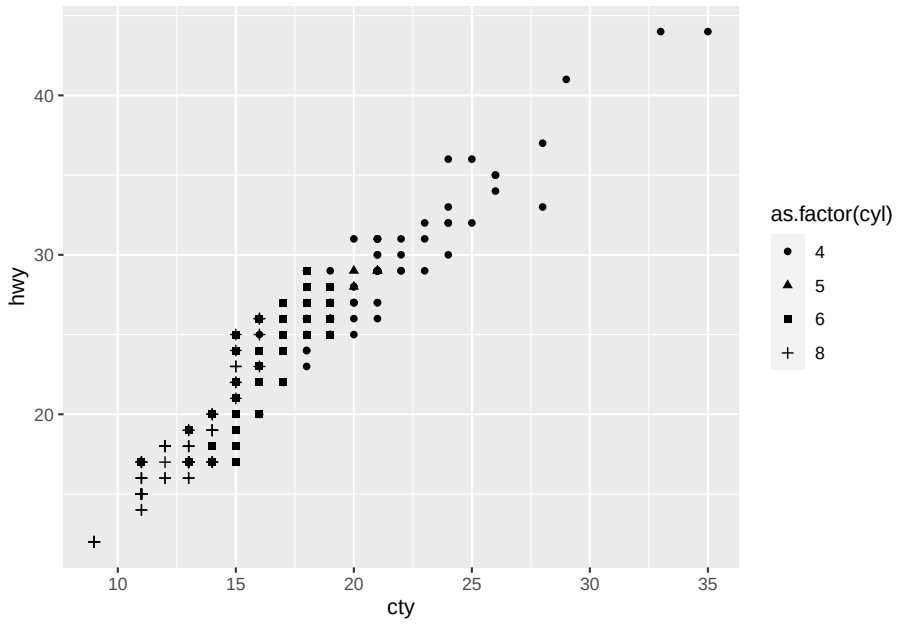
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



4.1.2 Formas

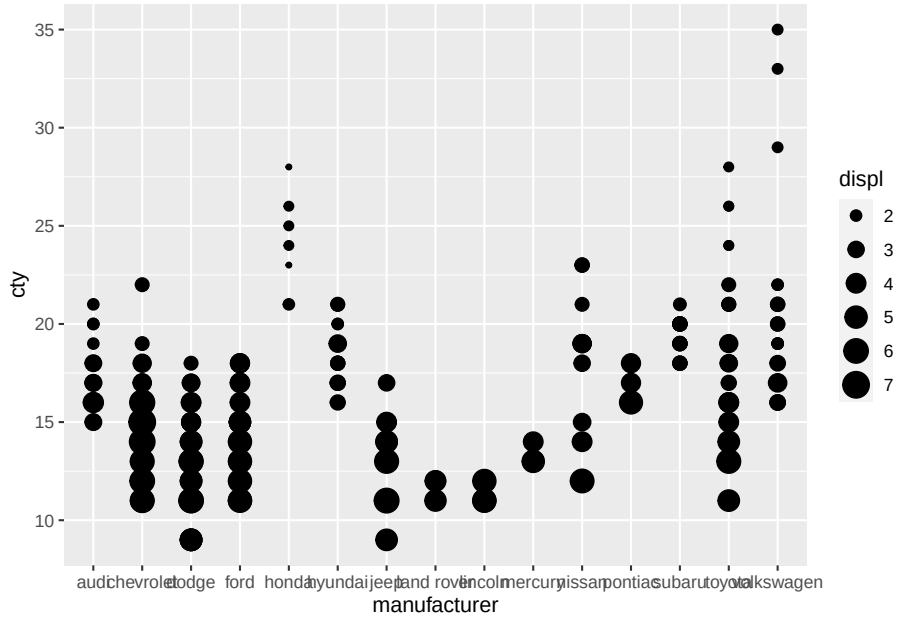
Al igual que con los colores, podemos personalizar las formas de nuestros gráficos (en particular de nuestros `geom_points()`). Para esto modificamos el argumento `shape`.

```
dispersion<-ggplot(mpg, aes(x=cty, y=hwy, shape=as.factor(cyl))) +
    geom_point()
dispersion
```



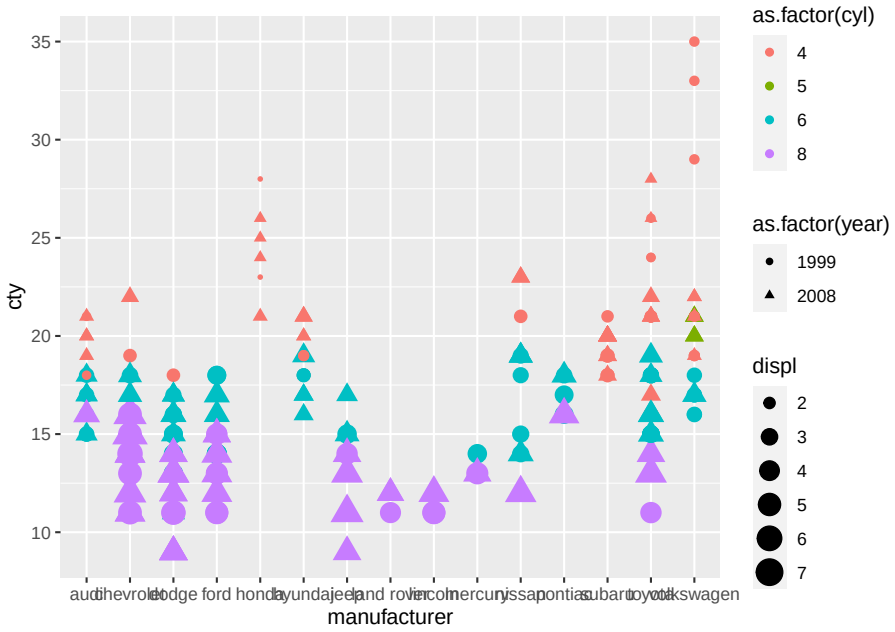

```
dispersion<-ggplot(mpg, aes(x=manufacturer, y=cty, size=displ)) +
    geom_point()
dispersion
```

4.1.3 Tamaño




```
dispersion<-ggplot(mpg, aes(x=manufacturer, y=cty, size=displ, shape=as.factor(year),c
  geom_point()
dispersion
```

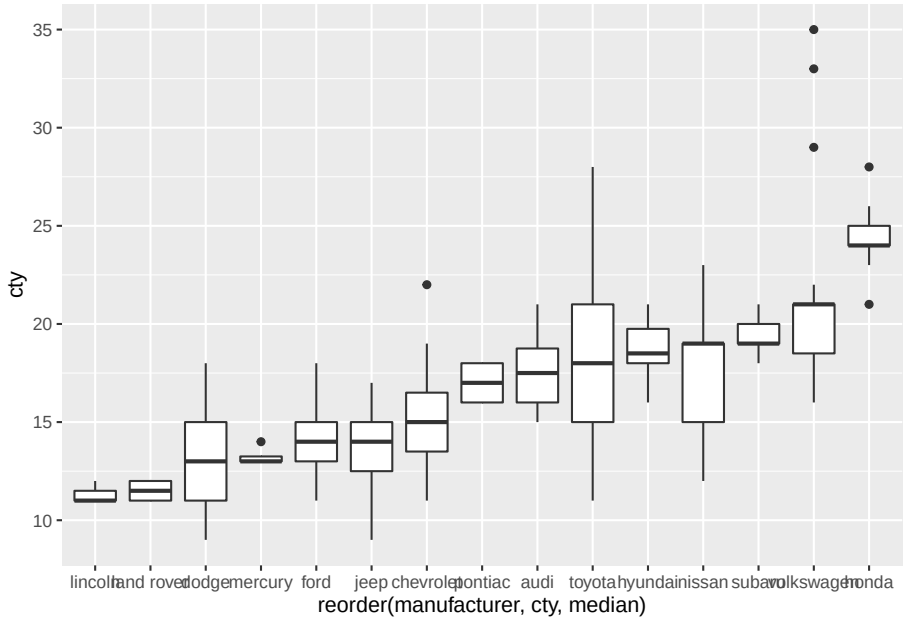
4.1.4 Mezclémoslas



4.1.5 Orden

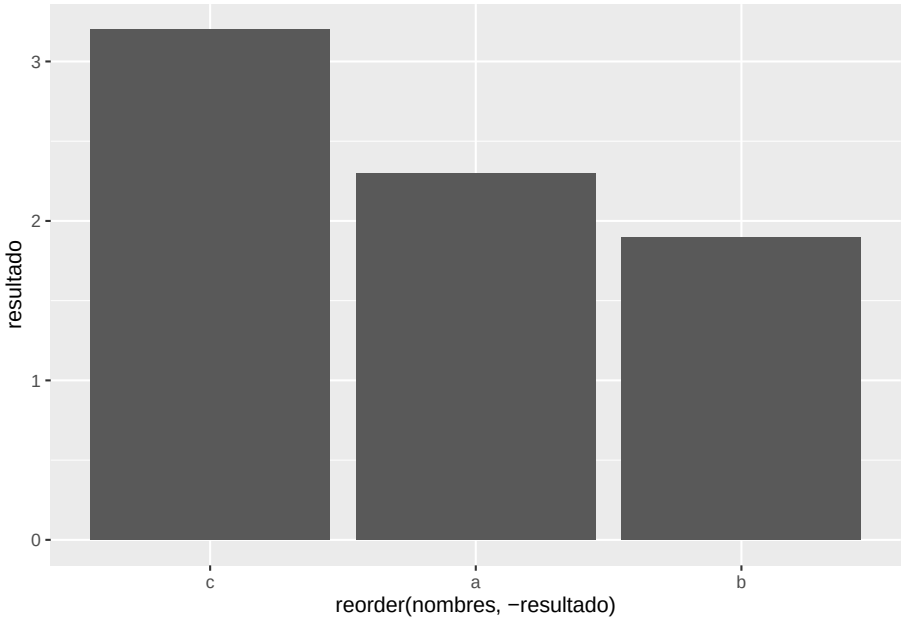
Podemos utilizar la función reorder() para

```
cajas<-ggplot(mpg, aes(x=reorder(manufacturer, cty, median), y=cty)) +
    geom_boxplot()
cajas
```



Para invertir el orden agregamos un guión ("-") antes del segundo argumento.

```
#Creamos datos de ejemplo
df <- data.frame(nombres = c("a", "b", "c"), resultado = c(2.3, 1.9, 3.2))
histograma <- ggplot(df, aes(x=reorder(nombres,-resultado), y=resultado)) +
  geom_col()
histograma
```


3
2
1
0
resultado
c
a
b
reorder(nombres, −resultado)