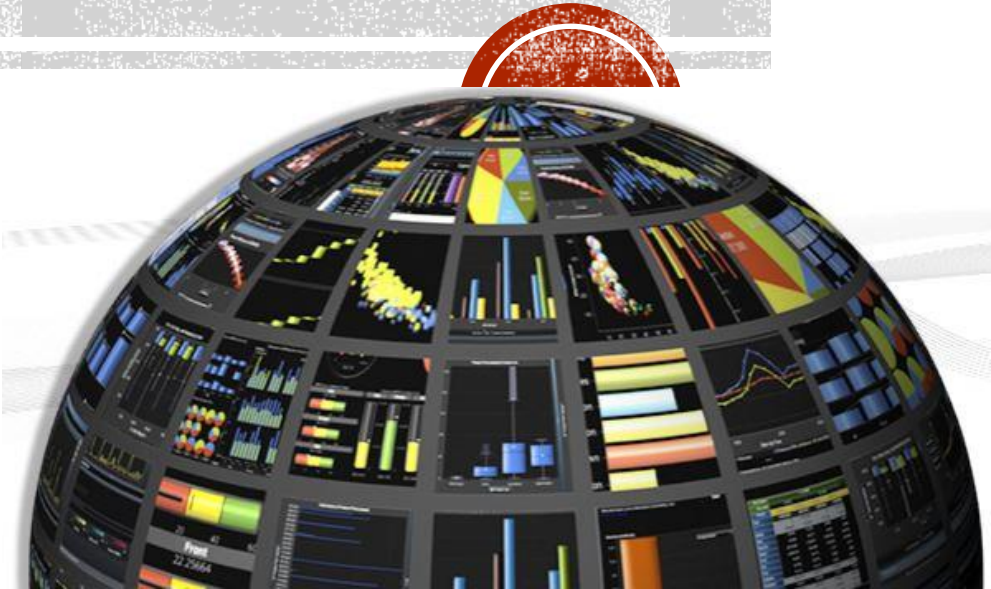
ANÁLISIS Y VISUALIZACIÓN DE CONJUNTOS DE DATOS

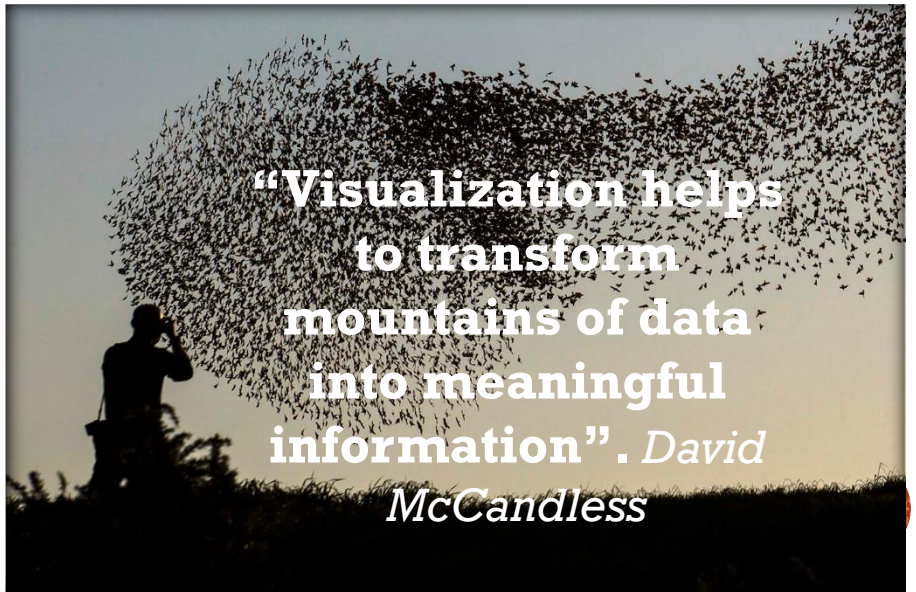
Christian Camilo Urcuqui López, MSc



UNIVERSIDAD ICESI | A OTRO NIVEL

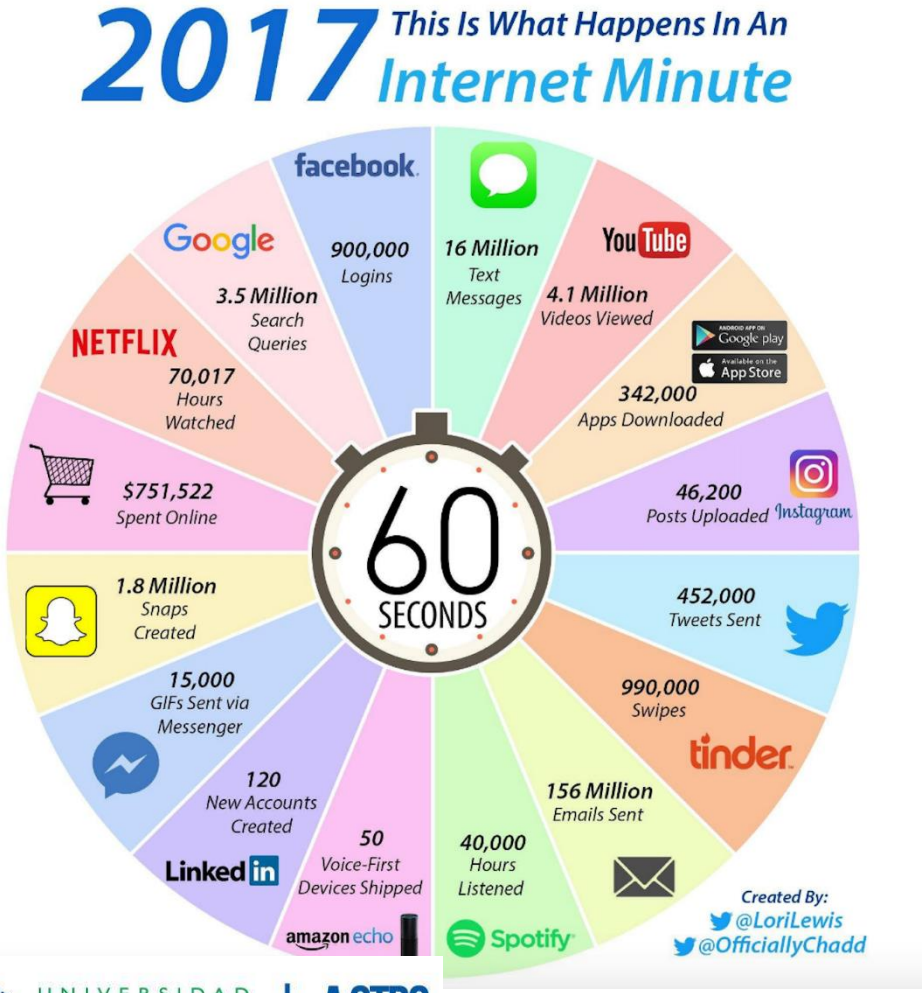
OBJETIVO E INSUMOS

- Aplicar técnicas de visualización para comunicar efectivamente los resultados de los modelos analíticos.
- **tidyverse,** es un colección de paquetes de R diseñados para ciencia de datos. install.packages ("tidyverse")
- **Treemap**. install.packages("treemap")
- **Openair**. install.packages("openair")
- **Quantmod.** install.packages ("quantmod")
- **Readxl.** install.packages("readxl")
- Descarguen el *Bike Sharing Dataset Data Set*



UNIVERSIDAD ICESI | A OTRO NIVEL

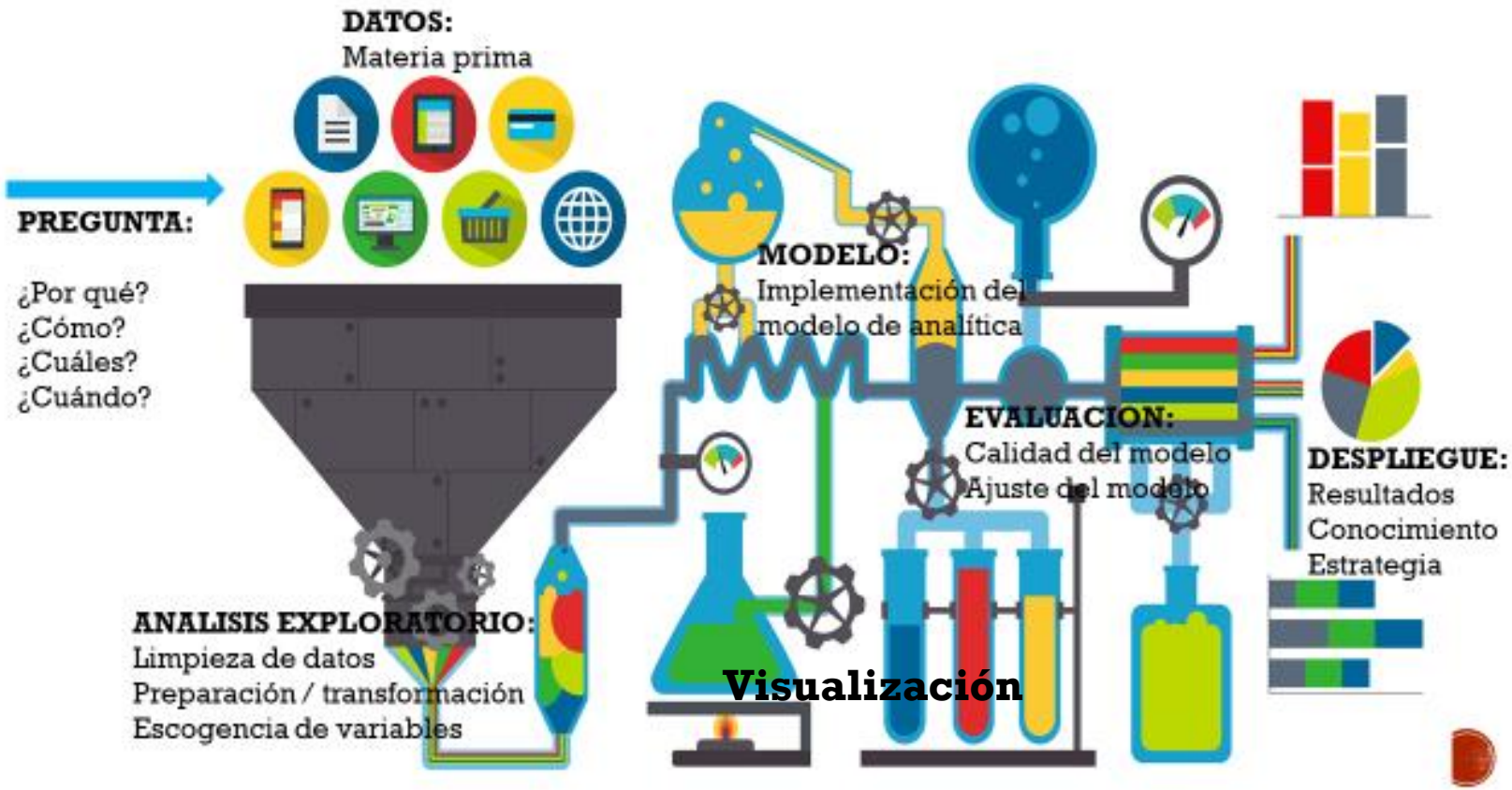
RECORDEMOS

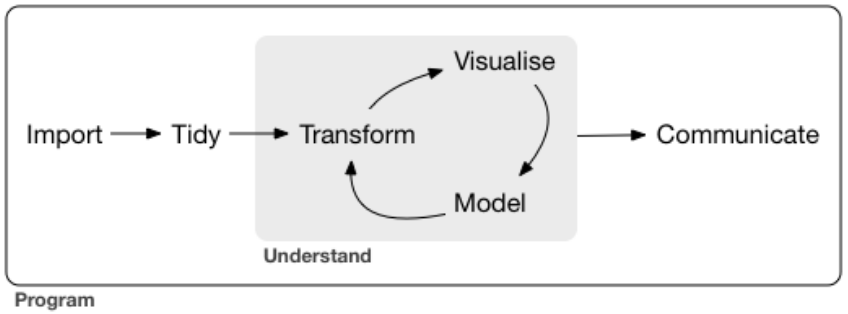


- **Volumen**
- **Velocidad**
- **Variedad**
- **Veracidad**
- Valor
- Variabilidad
- Visualización

UNIVERSIDAD ICESI | A OTRO NIVEL

RECORDEMOS





Marco de trabajo típico de un proyecto de ciencia de datos.
R for Data Science

VISUALIZACIÓN DE DATOS

- Es una técnica donde los resultados analíticos se comunican a través de insumos gráficos, por ejemplo, mapas, infografías, alertas y cuadrículas de datos.
- La visualización de datos tradicional provee en su mayoría resultados estáticos que incluyen gráficos, reportes y tableros. Mientras, que las visualizaciones contemporáneas son más interactivas y proveen tanto resúmenes y vistas más detalladas de los datos.

VISUALIZACIÓN DE DATOS

Características comunes en las herramientas de visualización en BI y Big Data:

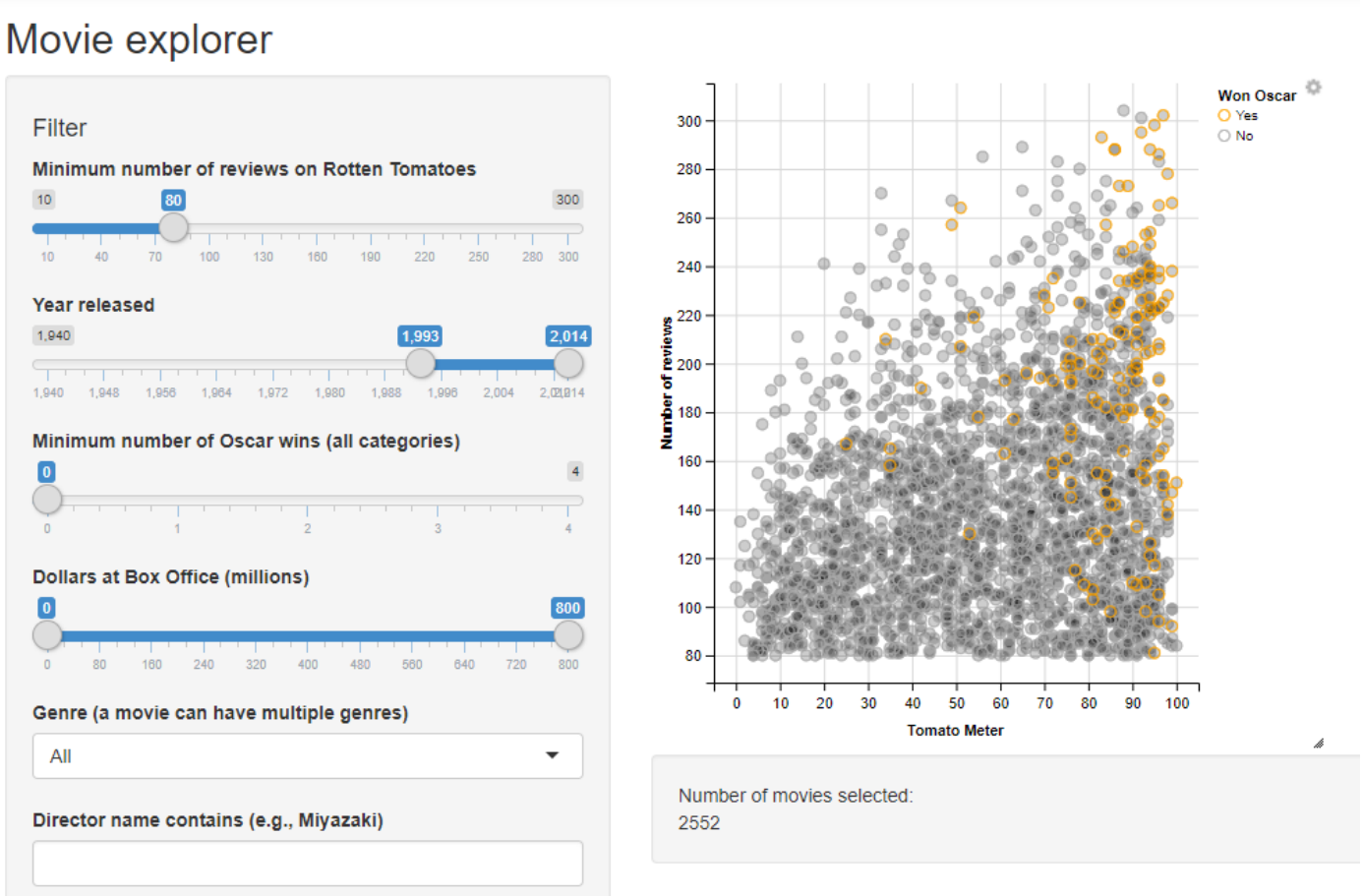
- Agregación: provee una visión holística y resumida a través de múltiples datos
- Drill- Down: permite una visión más detallada de los datos de interés focalizando a un solo subconjunto
- Filtrado: ayuda a analizar un conjunto particular de datos a través del filtro de información que no es de interés
- Roll-Up: Agrupa múltiples categorías de datos para mostrar subtotales y totales
- What-if Analysis: Ofrece distintos resultados para visualización a través de cambios dinámicos

VISUALIZACIÓN DE DATOS

Características de las herramientas de visualización avanzada

- Las herramientas de visualización en Big Data generalmente utilizan recursos de memoria para reducir la latencia.
- Se incorporan mecanismos para analítica predictiva, prescriptiva y transformación de datos.
- Permiten realizar análisis de datos no estructurados y semiestructurados.

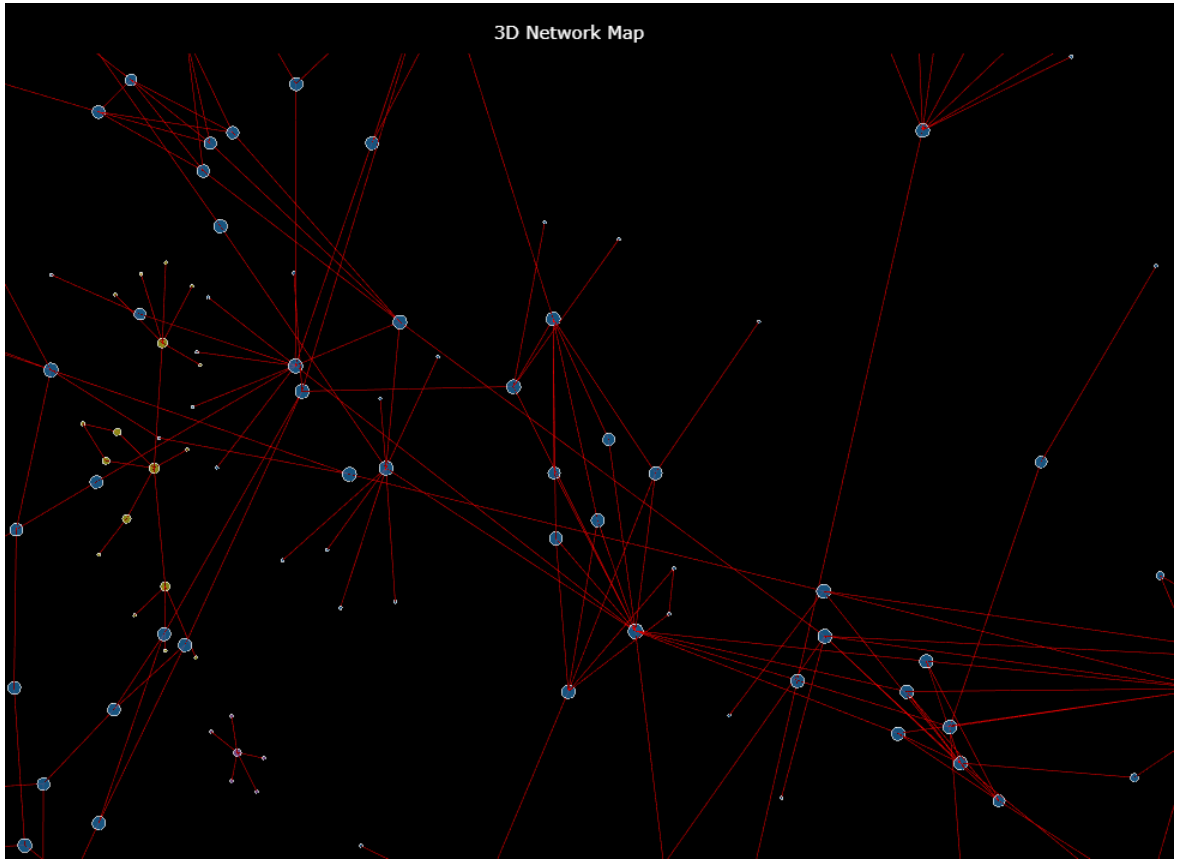
CASOS, DIAGRAMAS



Explorador de películas; elemento visual que integra un gráfico de dispersión y otros elementos interactivos que permiten visualizar distintas variables de las películas durante 1940-2014.

https://shiny.rstudio.com/gallery/movie-explorer.html

CASOS, REDES

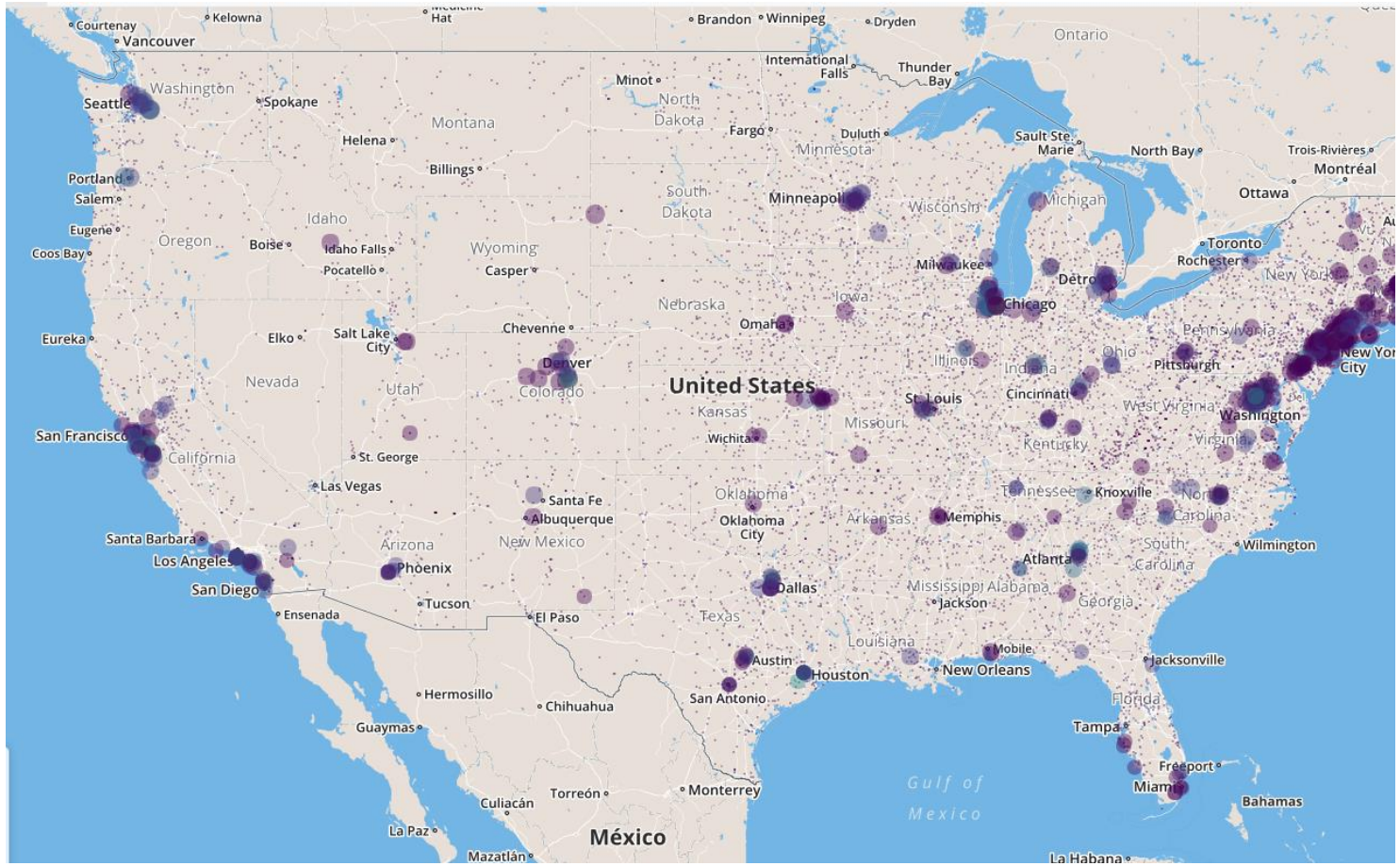


Mapa de red, este insumo gráfico nos ilustra las relaciones, los datos de cada nodo y los grupos que se generan entre los objetos.

https://gallery.shinyapps.io/3D_Mapping/

http://nodexlgraphgallery.org/Pages/Graph.aspx?graphID=145179

CASOS, MAPAS

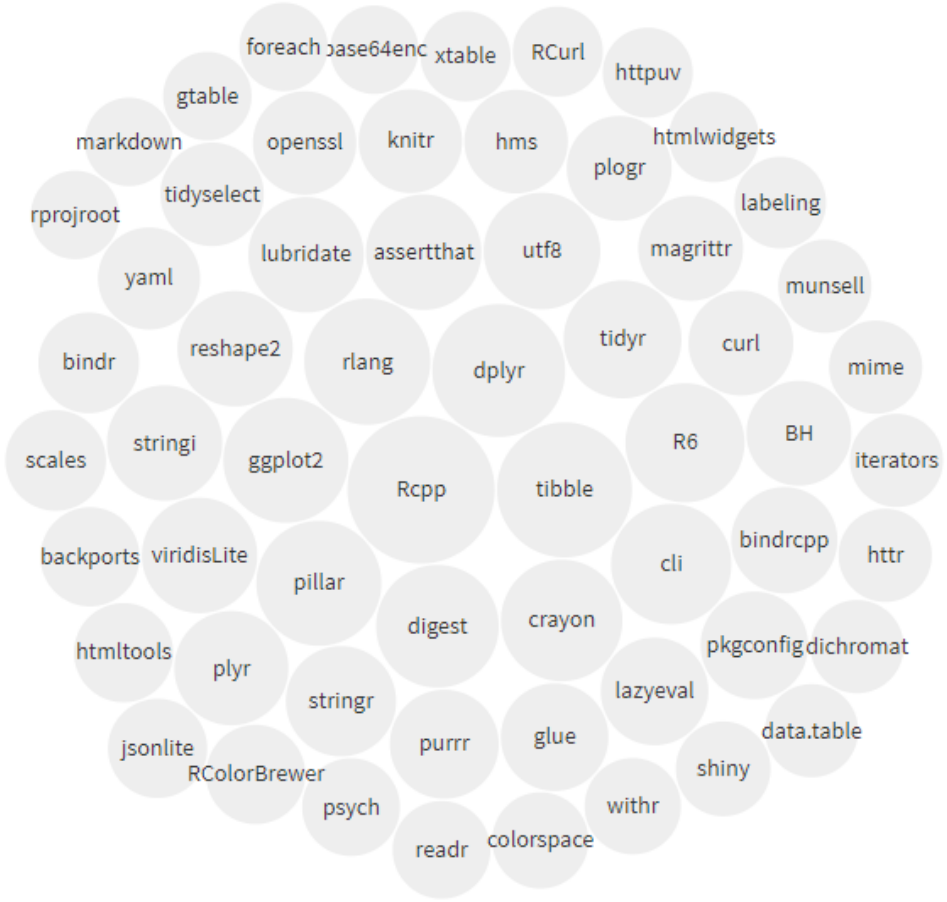


Mapa poblacional sobre el ranking Zip de USA. El Zip es un número entre 0 y 99 que representa el promedio de sus rangos de percentiles en educación universitaria y en los ingresos.

Data compiled for *Coming Apart: The State of White America, 1960–2010* by Charles Murray (Crown Forum, 2012)

https://shiny.rstudio.com/gallery/superzip-example.html

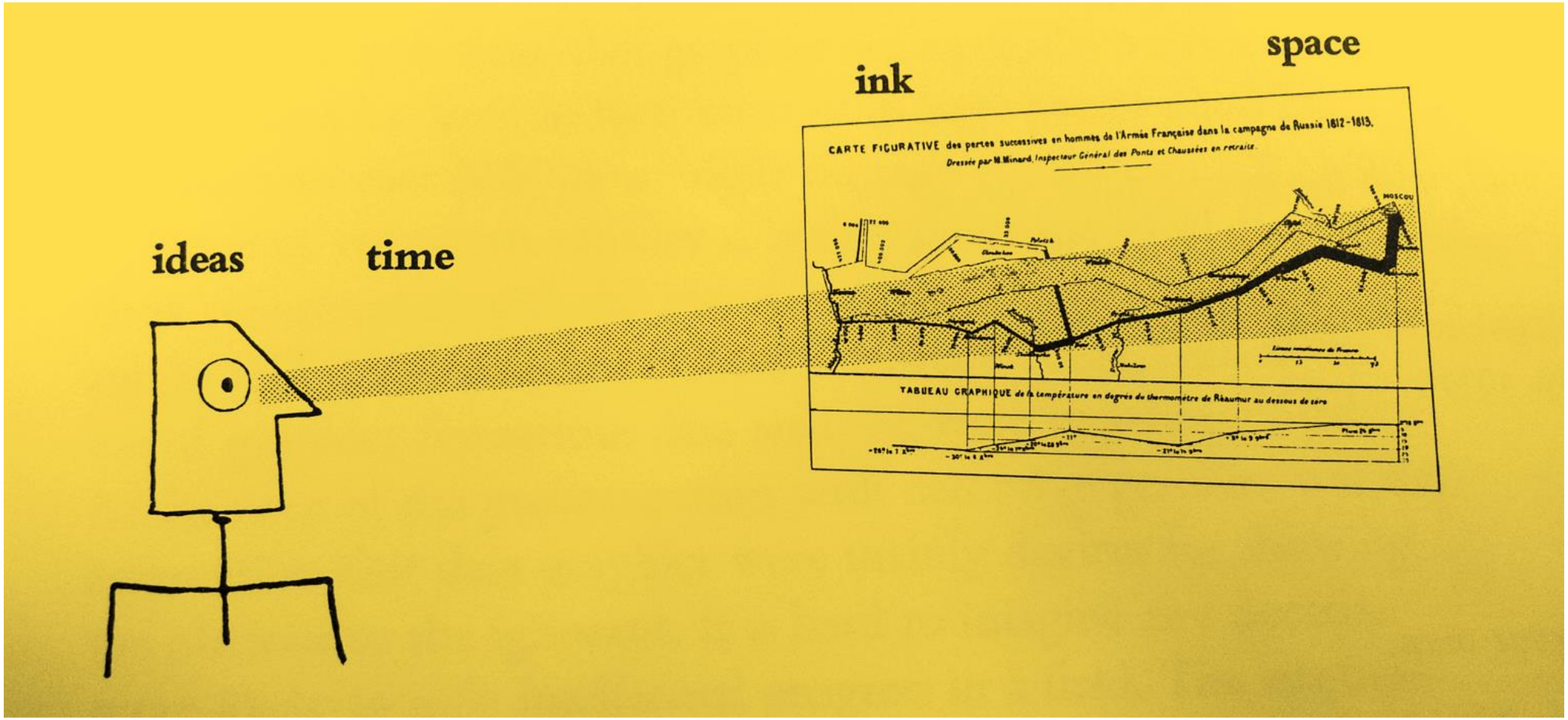
TIEMPO REAL



Este diagrama nos demuestra las capacidades de generar insumos visuales que recolectan y procesan datos en tiempo real, además, nos da la posibilidad de crear alarmas.

Los datos son los paquetes descargados de R.

https://gallery.shinyapps.io/087-crandash/



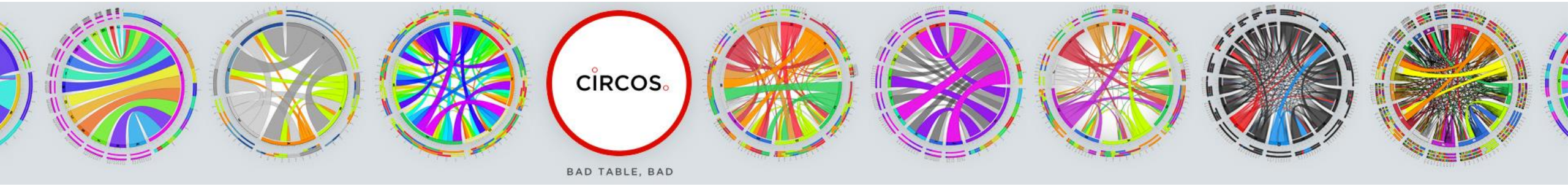
The visual display of quantitative information. E.Tufte,1998

ANÁLISIS EXPLORATORIO (EDA)

https://www.youtube.com/watch?v=BkSO9pOVpRM
https://youtu.be/jbkSRLYSojo

Tiene como objetivo el planteamiento de hipótesis y el descubrimiento de patrones o relaciones en la información.

TECNOLOGÍAS



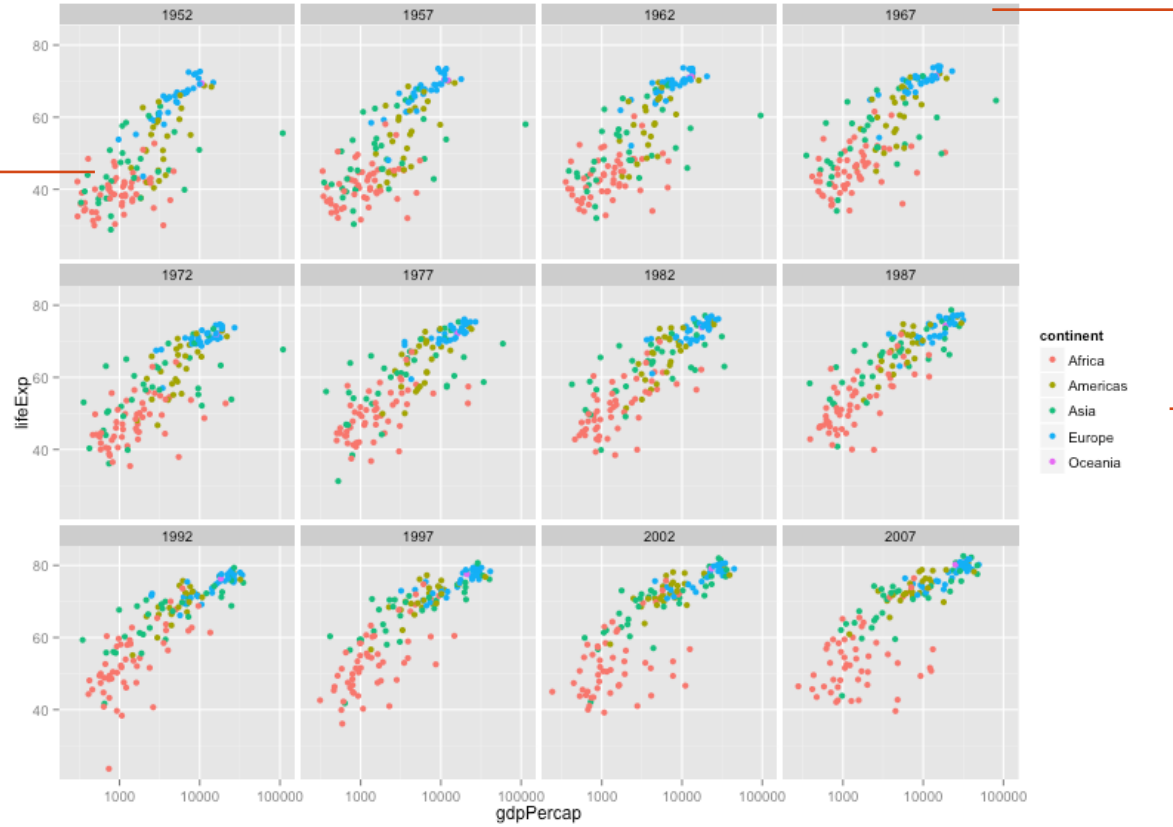
R

- **Ggplot2** es un paquete de R basado en la gramática de los gráficos (un sistema coherente para el desarrollo de gráficos) que permite desarrollar y definir visualizaciones de los datos de forma sencilla.
  - http://ggplot2.org/
- Adicionalmente, en R se pueden encontrar otras herramientas para la generación de reportes, por ejemplo, Markdown.
- Existen otros proyectos para el desarrollo de ilustraciones y el manejo de datos espaciales, por ejemplo Spatial.ly

GRAMÁTICA DE LOS GRÁFICOS ¿CUÁLES SON LOS COMPONENTES DE UN GRÁFICO?

Facets:
Grupos en los datos

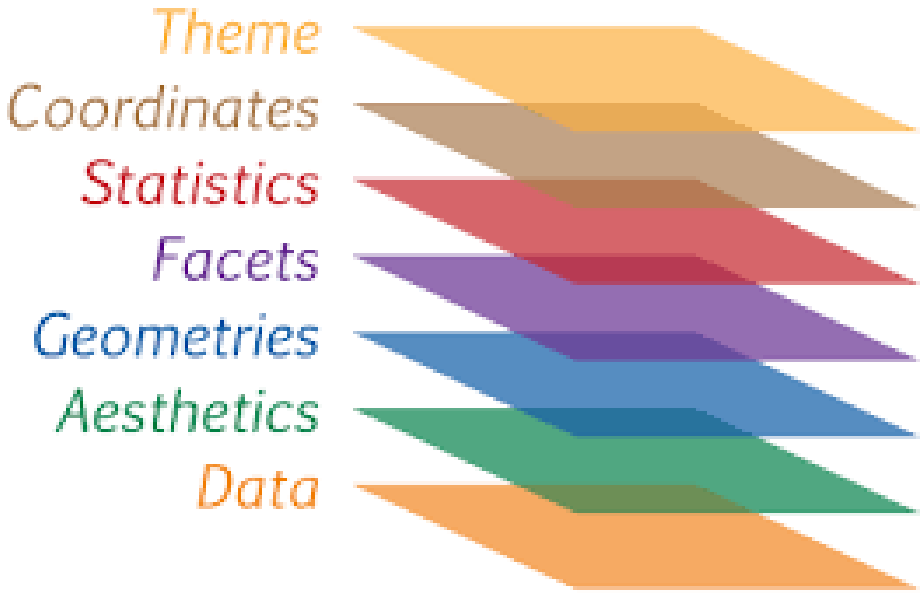
Geometry
Objeto geométrico
En este caso:
geom_point



Aesthetic
Papel que desempeña cada variable en el gráfico.
Por ejemplo:
- Continent: color
- gdpPercap: eje x
- LifeExp: eje y

Datos

GRAMÁTICA DE LOS GRÁFICOS

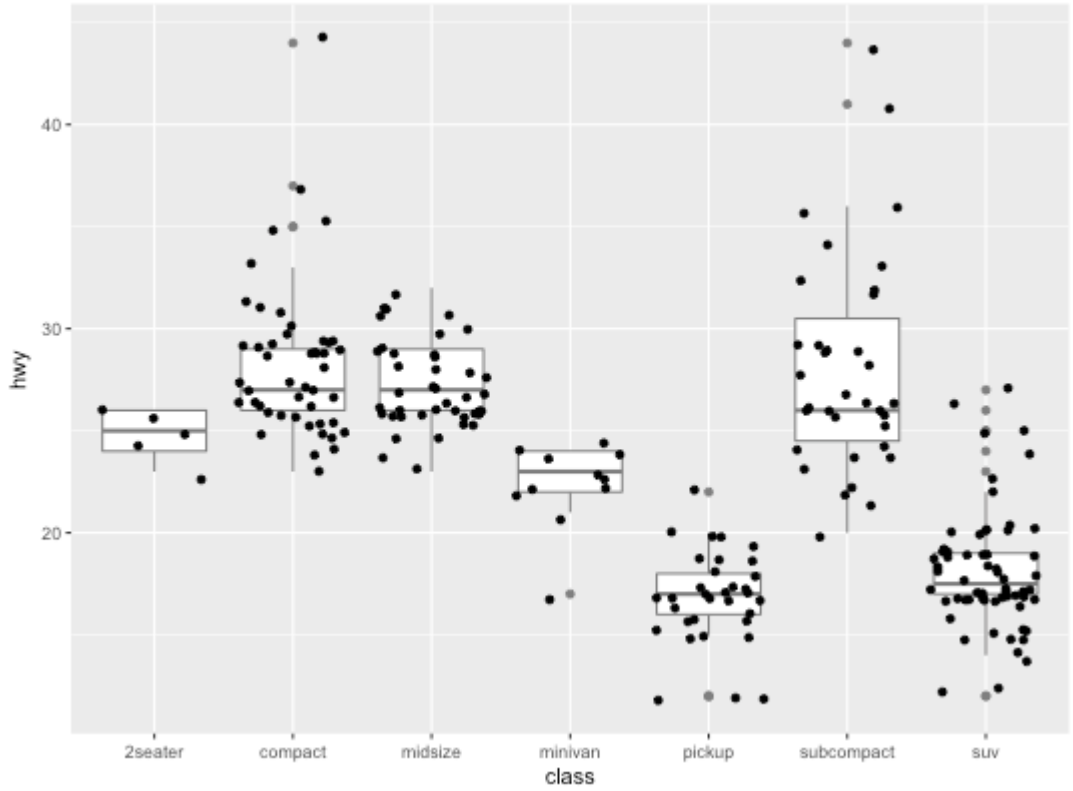


Por ejemplo:

```
ggplot(data=diamonds, aes(x=carat, y=price)) +
geom_point() +
 stat_smooth(method = lm) +
scale_x_log10() +
scale_y_log10()
```

GRAMÁTICA DE LOS GRÁFICOS

- Dado que ggplot2 maneja una gramática, eso permite varios tipos de gráficos en una misma figura. Por ejemplo, simultáneamente tener un grafico de dispersión y una caja de bigotes.



TIPOS DE VARIABLES

- **Variables cuantitativas**, sus valores son numéricos y pueden ser contados o medidos, por ejemplo, ventas netas de una compañía.
  - **Variables discretas**, es una variable numérica que usualmente se obtiene a través del conteo y solamente puede tomar valores específicos de un conjunto, por ejemplo, el número de personas en una ciudad o el número de quejas de los clientes.
  - **Variables continuas**, son variables numéricas que pueden tomar un valor (infinito/decimal) entre dos valores numéricos cualquiera. Usualmente, esta variable se obtiene a partir de mediciones, por ejemplo, la temperatura de un paciente.

TIPOS DE VARIABLES

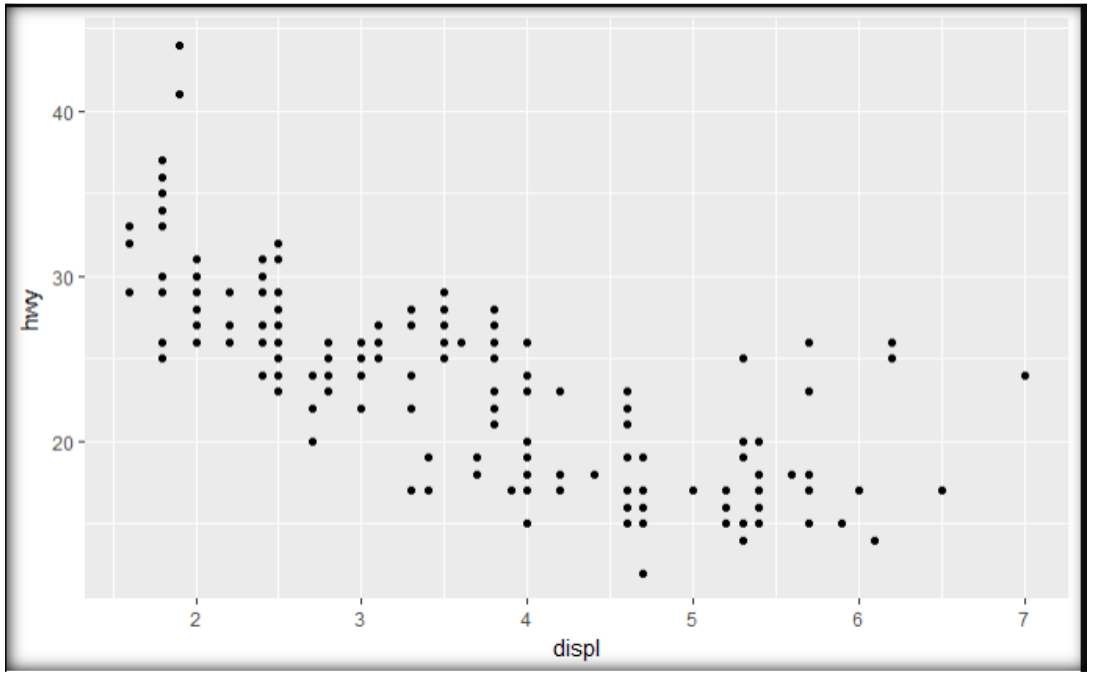
- **Variables cualitativas**, conocidos también como variables categóricas, sus valores pueden ser contados pero no medidos.
  - **Variables nominales**, son valores que presentan a una categoría y no cuentan con un orden. Estos valores pueden ser contados pero no pueden ser ni medidos y ni ordenados, por ejemplo, género de música y categorías de productos.
  - **Variables ordinales**, son valores numéricos que pueden ser discretos o continuos y que están ya sea ordenadas o jerarquizadas.
  - **Variables binarias**, sus valores hacen parte únicamente a dos categorías que generalmente son opuestos, por ejemplo, 1/0 y verdadero/falso.

TIPOS DE VARIABLES

- **Variables independientes**, sus valores no dependen de otra variable pero posiblemente si puedan influenciar a otras.
- **Variables dependientes**, este tipo de variable si depende de otras variables.
- **Variable aleatoria**, es una variable que puede asumir un valor de un rango de valores basado en la probabilidad.

TIPOS DE GRÁFICOS

- **Gráfico de dispersión (Scatter Plot)**, permite ver la asociación entre dos variables, además, ayuda al descubrimiento de patrones y datos atípicos. Usualmente, es utilizado para el análisis de correlación y de regresión.



CASO 1, DATASET DE CARROS

- Para ilustrar las capacidades de Ggplot2 procederemos a instalar el paquete tidyverse. Esta librería contiene un conjunto de paquetes para la aplicación de la ciencia de datos, por ejemplo:

```
> library(tidyverse)
-- Attaching packages --------------- tidyverse 1.2.1 --
v ggplot2 2.2.1     v purrr   0.2.4
v tibble  1.4.2     v dplyr   0.7.4
v tidyr   0.8.0     v stringr 1.2.0
v readr   1.1.1     v forcats 0.3.0
```

- Vamos a utilizar el *dataset* ***mpg*** que contiene información acerca de 38 modelos de carros recolectados por la US Environment Protection Agency entre 1999-2008.

```
> mpg
# A tibble: 234 x 11
   manufacturer model   displ  year   cyl trans   drv
   <chr>        <chr>   <dbl> <int> <int> <chr>   <chr>
 1 audi         a4       1.80  1999     4 auto(l~ f
 2 audi         a4       1.80  1999     4 manual~ f
 3 audi         a4       2.00  2008     4 manual~ f
 4 audi         a4       2.00  2008     4 auto(a~ f
```

CASO 1, EXPLORACIÓN Y DICCIONARIO

- Exploremos los datos, ¿Qué variables hay en el *dataset* y de que tipo son?

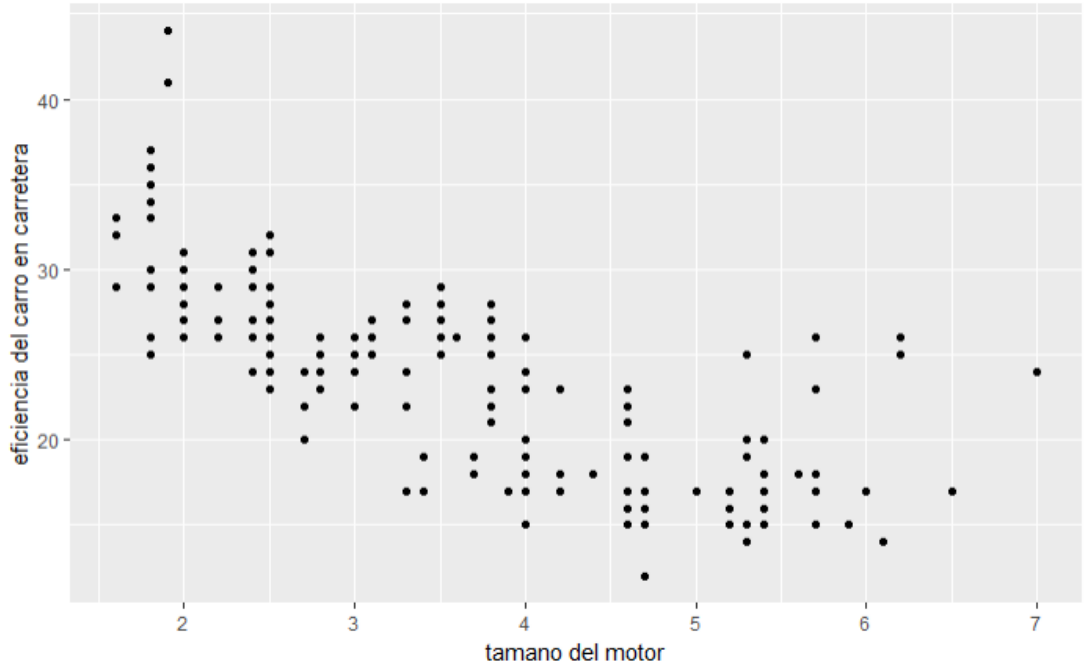
```
> str(mpg)
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':       234 obs. of  11 variables:
 $ manufacturer: chr  "audi" "audi" "audi" "audi" ...
 $ model       : chr  "a4" "a4" "a4" "a4" ...
 $ displ       : num  1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...
 $ year        : int  1999 1999 2008 2008 1999 1999 2008 1999 1999 2008 ...
 $ cyl         : int  4 4 4 4 6 6 6 4 4 4 ...
 $ trans       : chr  "auto(l5)" "manual(m5)" "manual(m6)" "auto(av)" ...
 $ drv         : chr  "f" "f" "f" "f" ...
 $ cty         : int  18 21 20 21 16 18 18 18 16 20 ...
 $ hwy         : int  29 29 31 30 26 26 27 26 25 28 ...
 $ fl          : chr  "p" "p" "p" "p" ...
 $ class       : chr  "compact" "compact" "compact" "compact" ...
```

  - *displ*, es el atributo que representa al tamaño del motor (capacidad de desplazamiento), los datos están en litros
  - *hwy,* es la variable de la eficiencia del carro en carretera, los datos están millas por galón (mpg)
  - No olvidemos que existe la opción de ayuda; la descripción de los otras variables se puede obtener con ?mpg

CASO 1, HELLO PLOT

- Vamos a utilizar un primer gráfico para responder las siguientes preguntas:
  - ¿Los carros con motor más grande consumen más gasolina que aquellos con motor más pequeño?
  - ¿Cuál es la relación entre el tamaño del motor y la eficiencia del combustible?

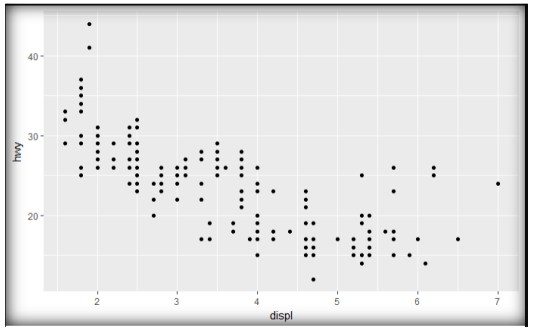
```
ggplot(data=mpg) +
  geom_point(mapping = aes(x=displ, y=hwy)) +
  xlab("tamano del motor") +
  ylab("eficiencia del carro en carretera")
```



GGPLOT

- Iniciamos nuestro plot a través de la función **ggplot()**, este nos permite crear un sistema de coordenadas para la adición de capas (*layers*).
- En nuestro previo caso la capa que se adiciono fue un diagrama de dispersión (*Scatterplot*) a través de la función **geom_point**.
- **Mapping** es un argumento que va acompañado de **aes** el cual nos permite asociar las variables de nuestro *dataset* a los ejes x y y

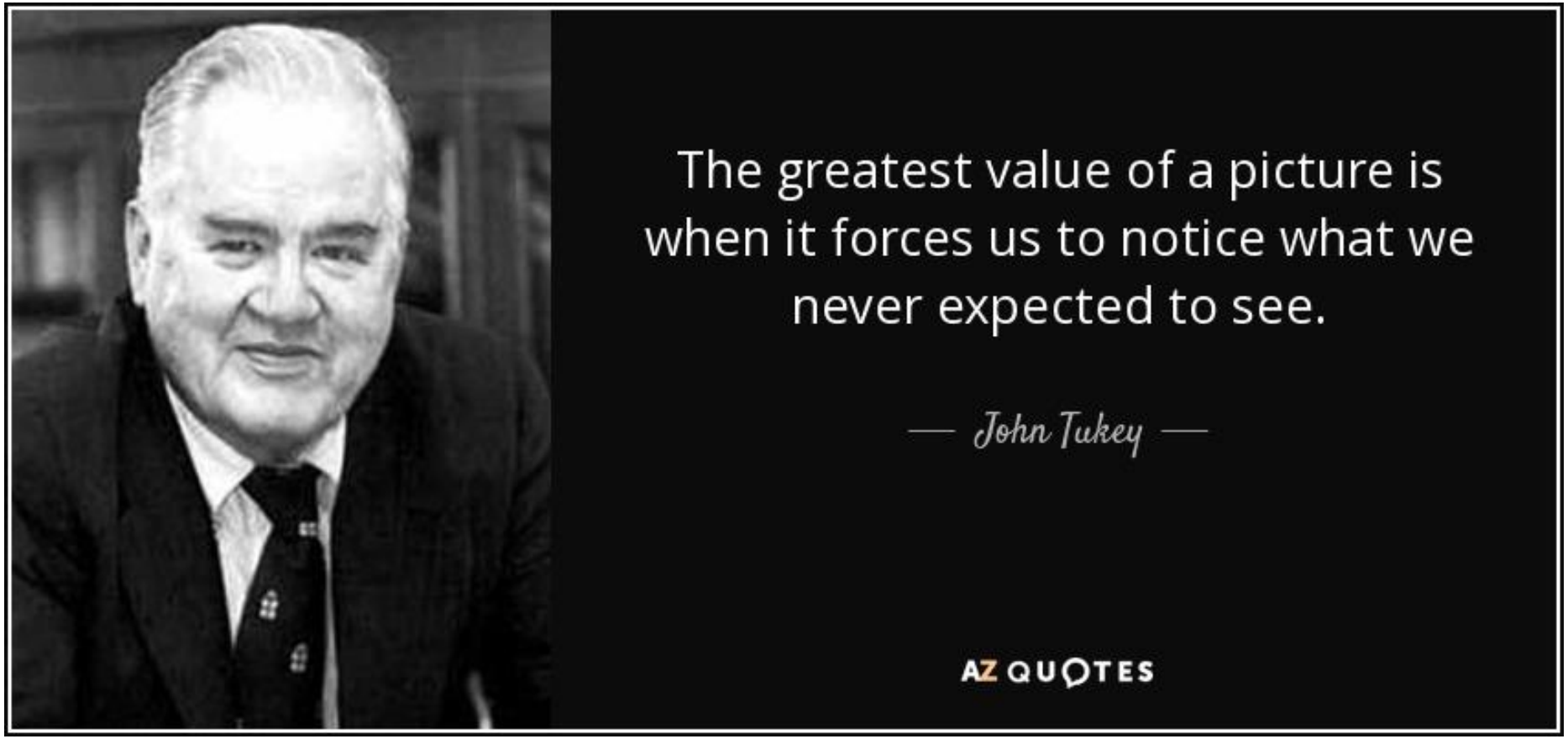
```
ggplot(data=mpg) +
  geom_point(mapping = aes(x=displ, y=hwy))
```



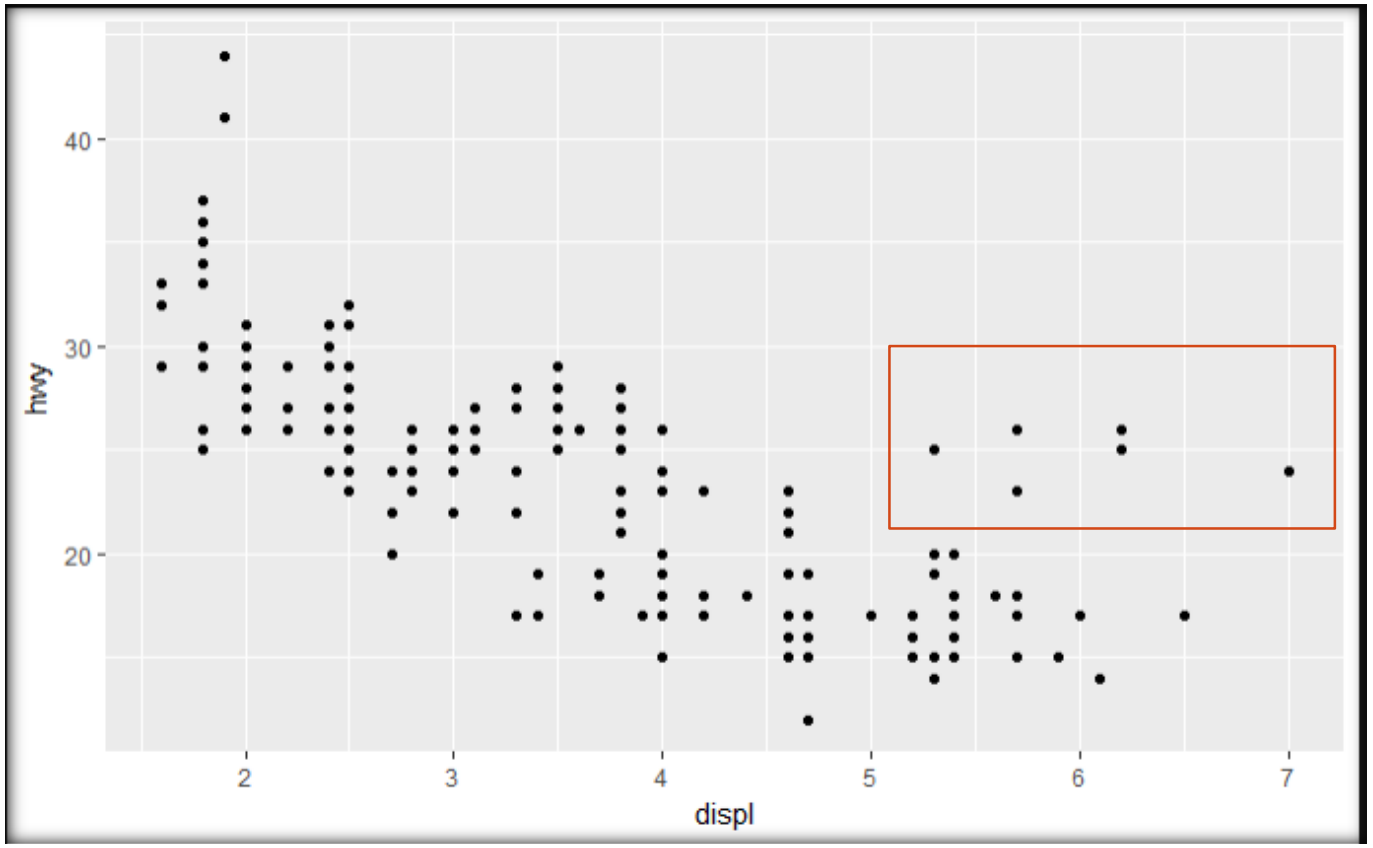

```
ggplot(data= <Dataset> ) +
  <funcion del grafico> mapping = aes (<mapeo de las variables>)
```

EJERCICIO 1. CARROS

- Del conjunto *mpg* haga un diagrama de dispersión de las variables *hwy* vs *cyl.*
  - *¿Qué puede observar en el diagrama?*
- Del conjunto *mpg* haga un diagrama de dispersión de las variables *class* vs *drv.*
  - *¿Qué puede inferir de estos gráficos? ¿Por qué no son funcionales o usables estos insumos?*


The greatest value of a picture is when it forces us to notice what we never expected to see.
— John Tukey —
AZ QUOTES

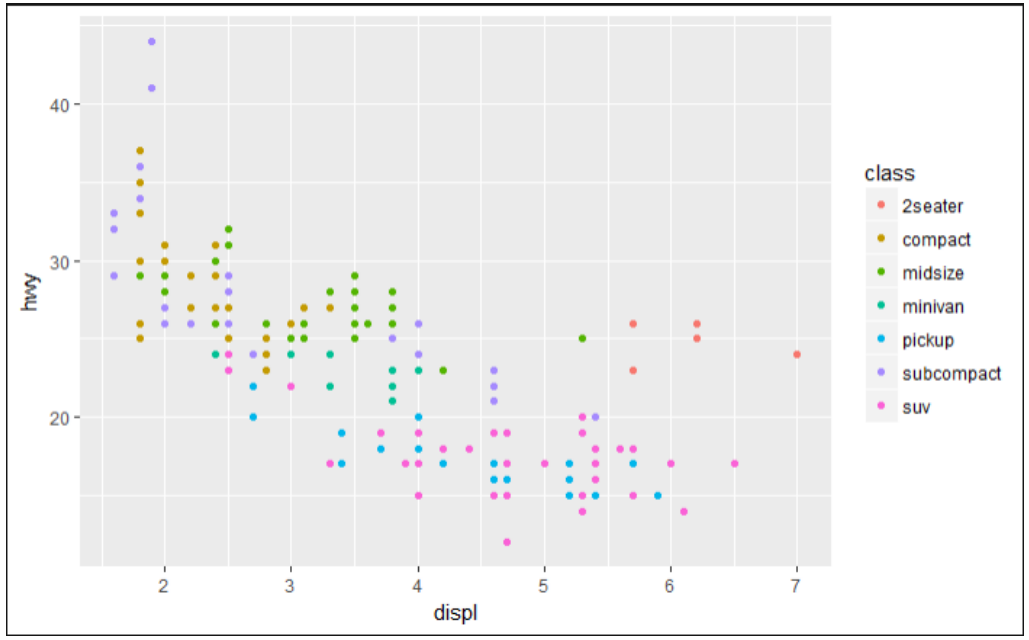
EJERCICIO 1. CARROS



EJERCICIO 1. CARROS

- ¿Ese grupo de datos tiene alguna particularidad que los asocia? ¿Existe una variable que nos ayude a identificarlos mejor?
- Podemos agregar una tercera variable (por ejemplo, *class*) a nuestro anterior diagrama de dispersión de dos dimensiones a través de la propiedad *aesthetic. Aesthetic* nos permite adicionar cambios en nuestros puntos, por ejemplo, agregar color, tamaño y forma.

```
ggplot(data=mpg) +
  geom_point(mapping=aes(x=displ, y=hwy, color=class))
```



EJERCICIO 1. CARROS

- No olvidemos revisar la opción de ayuda de cada función, si ingresamos a la información de **geom_point**, podemos encontrar lo siguiente:

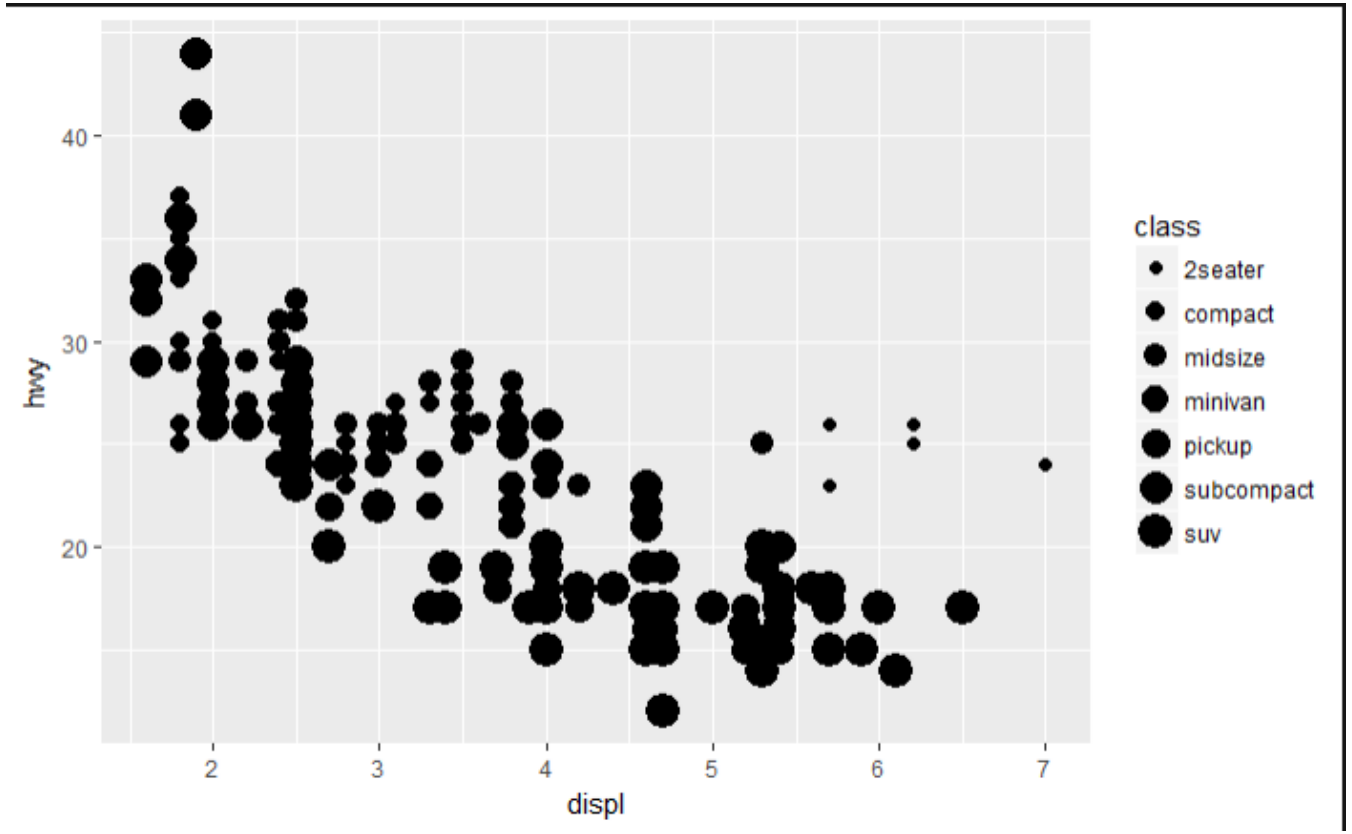
Aesthetics

`geom_point` understands the following aesthetics (required aesthetics are in bold):

- **`x`**
- **`y`**
- `alpha`
- `colour`
- `fill`
- `group`
- `shape`
- `size`
- `stroke`

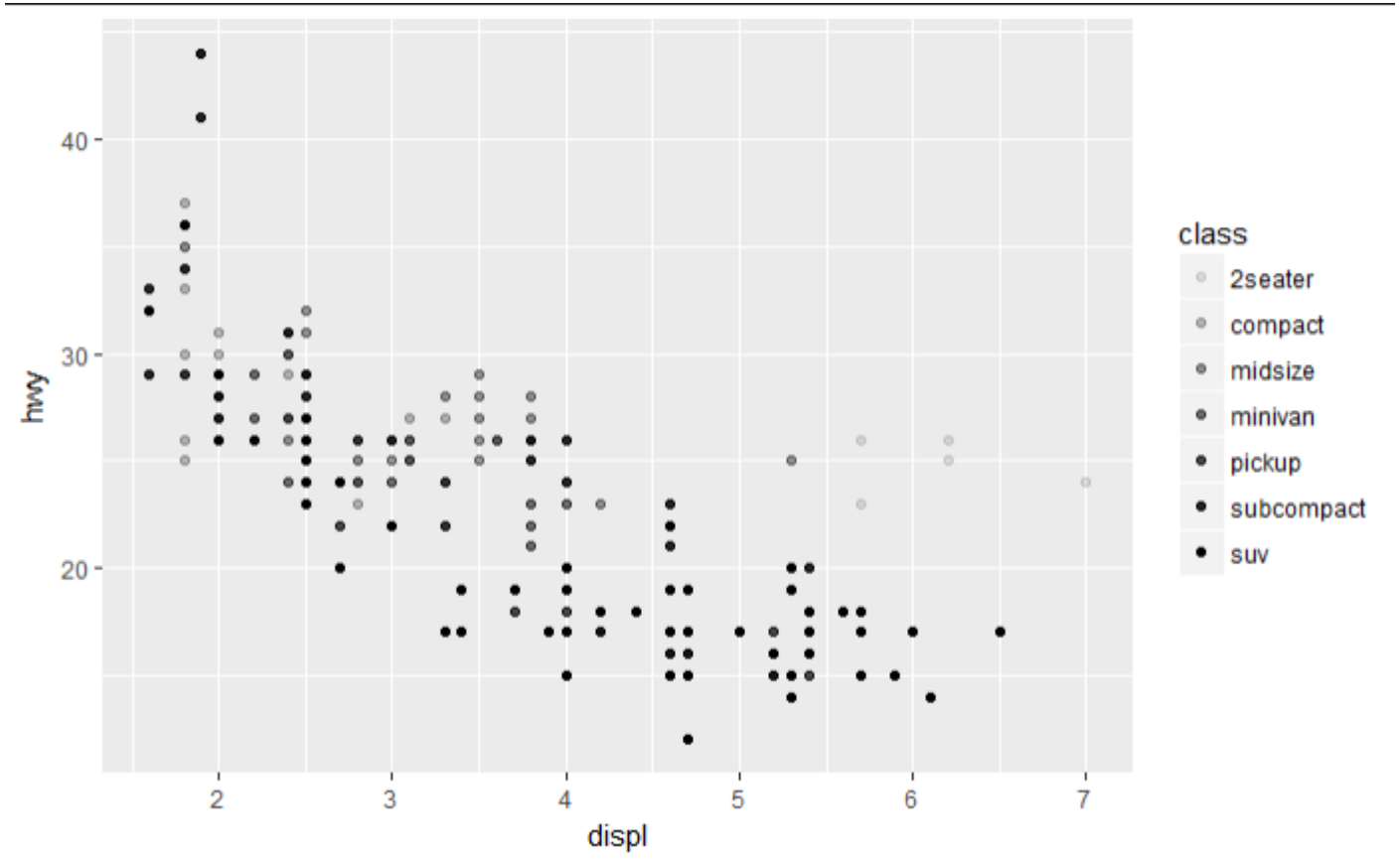
EJERCICIO 1. CARROS

```
ggplot(data=mpg) +
  geom_point(mapping = aes(x=displ, y=hwy, size=class))
```



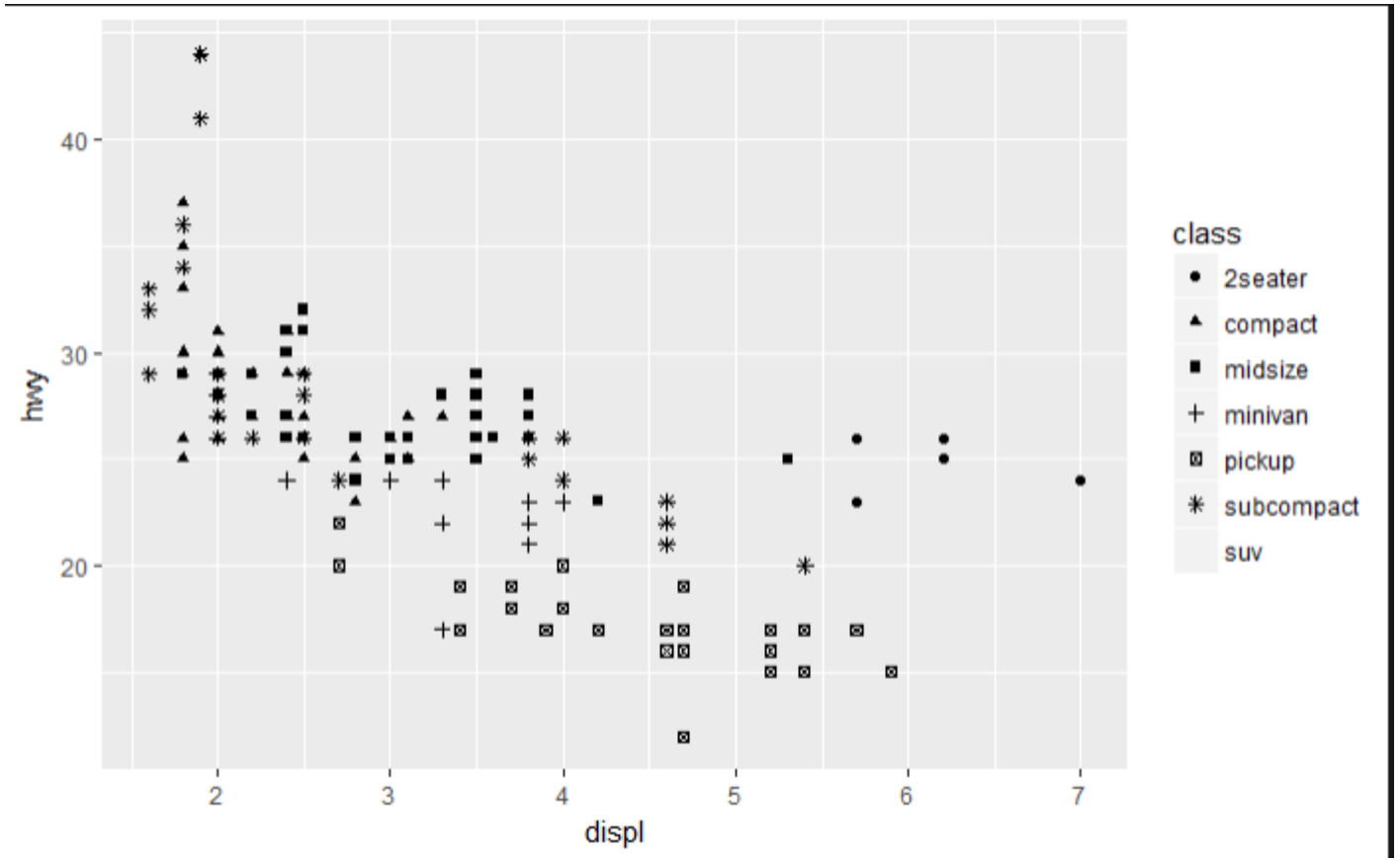
EJERCICIO 1. CARROS

```
ggplot(data=mpg) +
  geom_point(mapping = aes(x=displ, y=hwy, alpha=class))
```



EJERCICIO 1. CARROS

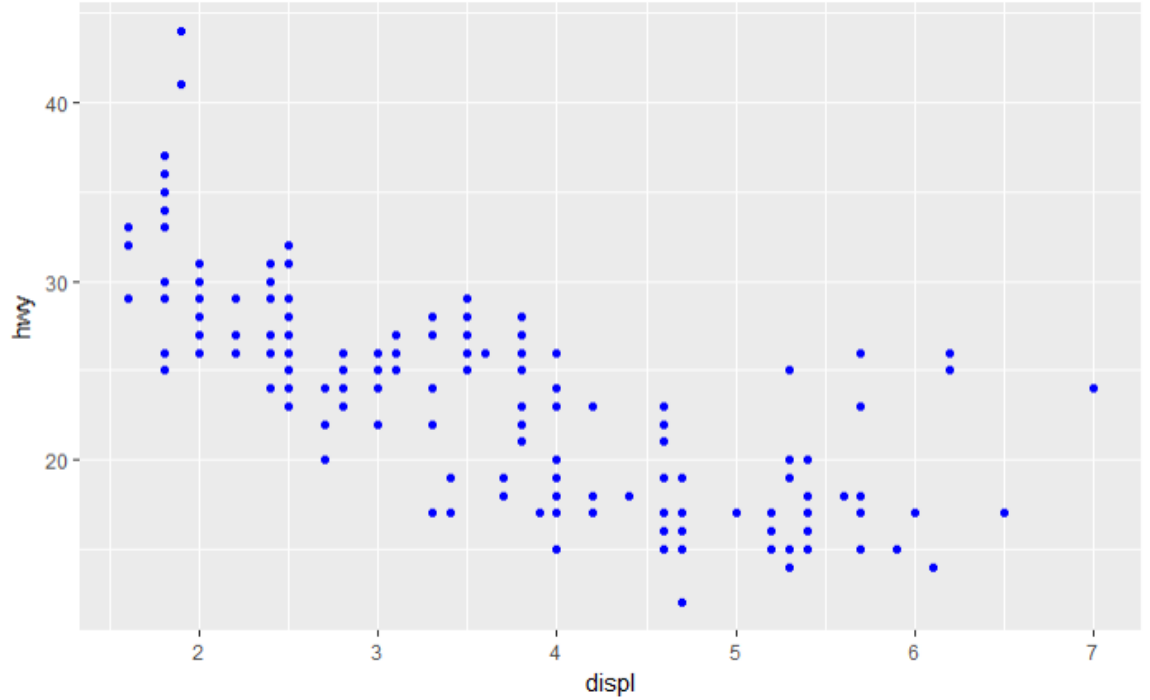
```
ggplot(data=mpg) +
  geom_point(mapping = aes(x=displ, y=hwy, shape=class))
```



EJERCICIO 1. CARROS

- Es posible editar las propiedades por defecto de *Aesthethic*, por ejemplo, cambiemos las propiedades del color para que no sea dependiente de los valores categóricos asignados a una variable.

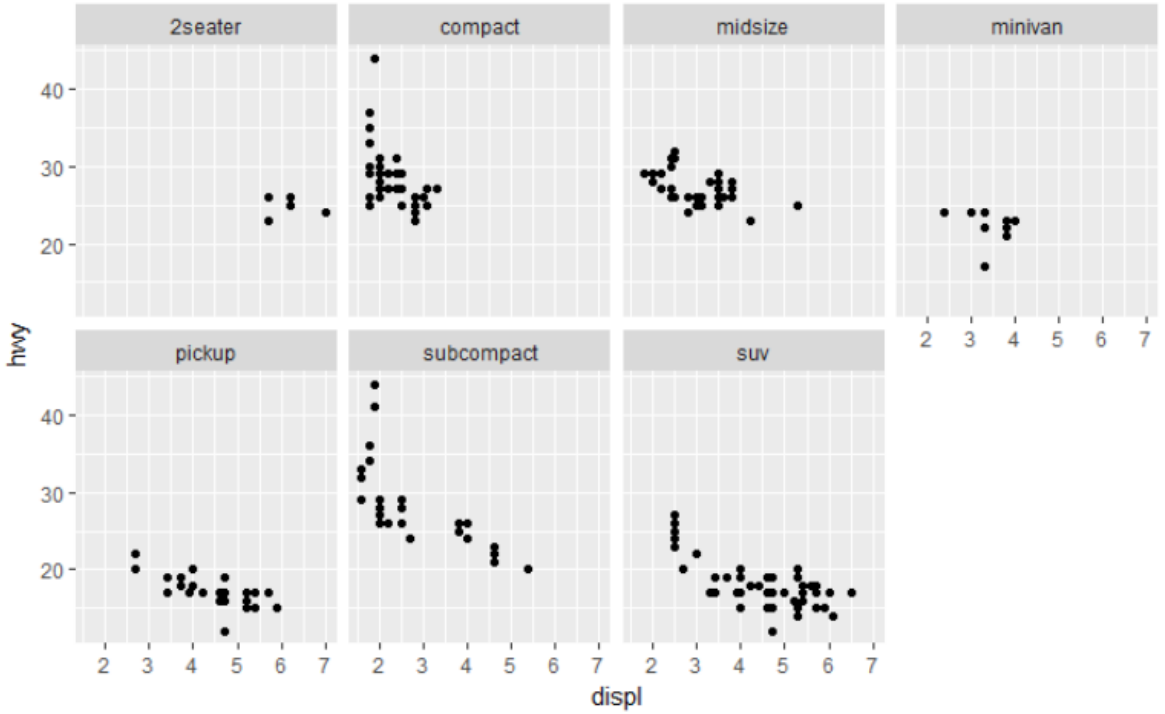
```
ggplot(data=mpg) +
  geom_point(mapping=aes(x=displ, y=hwy), color="blue")
```



FACETS

- Otra facilidad que existe en R es la posibilidad de dividir nuestro *plot* en diferentes *facets* a partir de una variable categórica. La ~ representa una formula, específicamente, una estructura de datos en R.

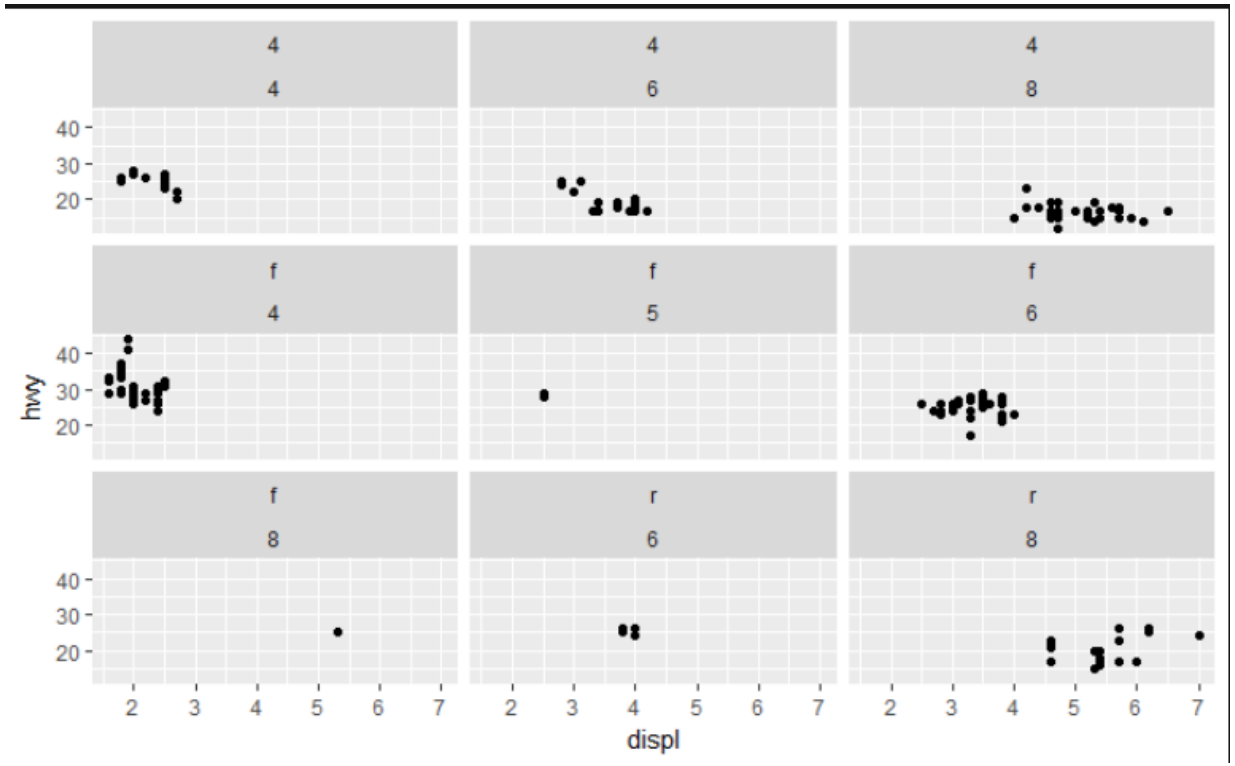
```
ggplot(data=mpg) +
  geom_point(mapping=aes(x=displ, y=hwy)) +
  facet_wrap(~ class, nrow=2)
```



FACETS

- Para adicionar más de una variable a la formula.

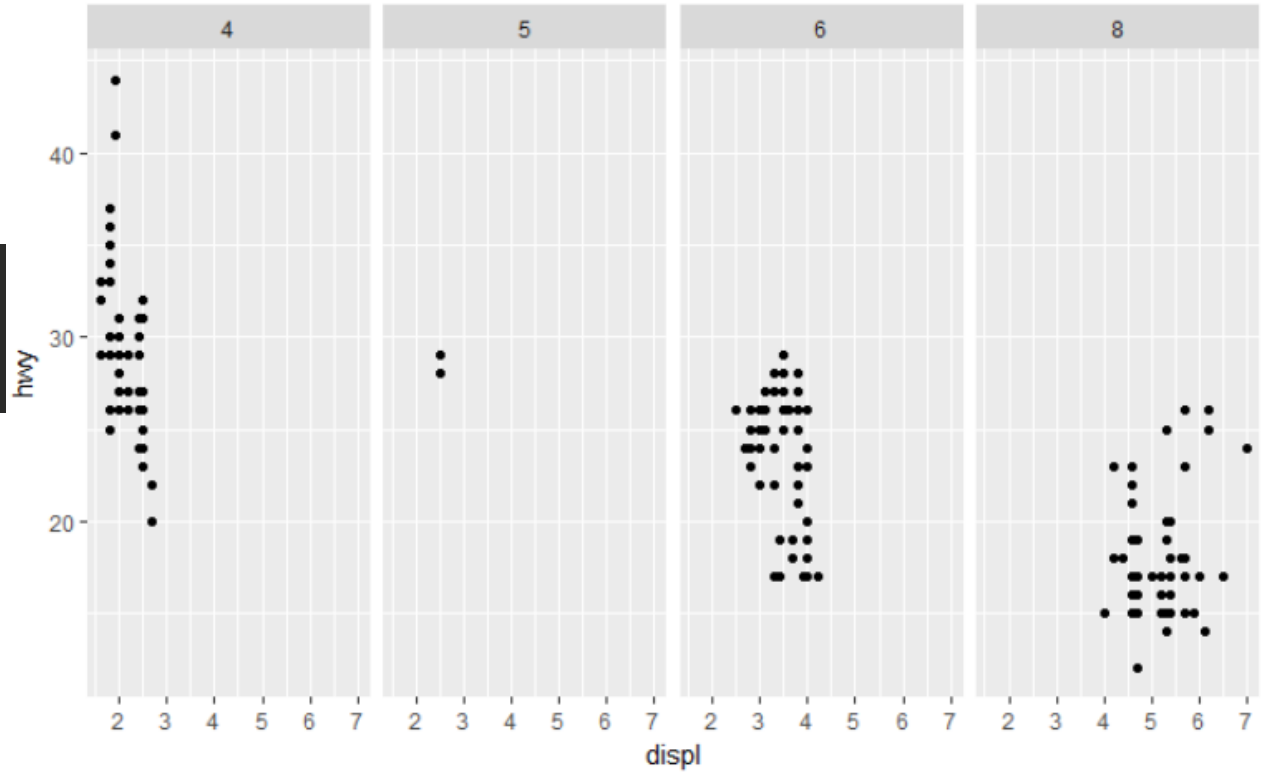
```
ggplot(data=mpg) +
  geom_point(mapping=aes(x=displ, y=hwy)) +
  facet_wrap(drv ~ cyl)
```



FACETS

- Si no quisiéramos tener *facets* dimensionados en filas o columnas

```
ggplot(data=mpg) +
  geom_point(mapping=aes(x=displ, y=hwy)) +
  facet_grid(. ~ cyl)
```



EJERCICIO 2. FACETS

- ¿Qué sucede con nuestro *facet* si utilizamos una variable continua?
- ¿Por qué hay celdas vacías en el *plot* con facet_grid(drv ~ cyl)? ¿Cómo esta relacionado con este *plot*?

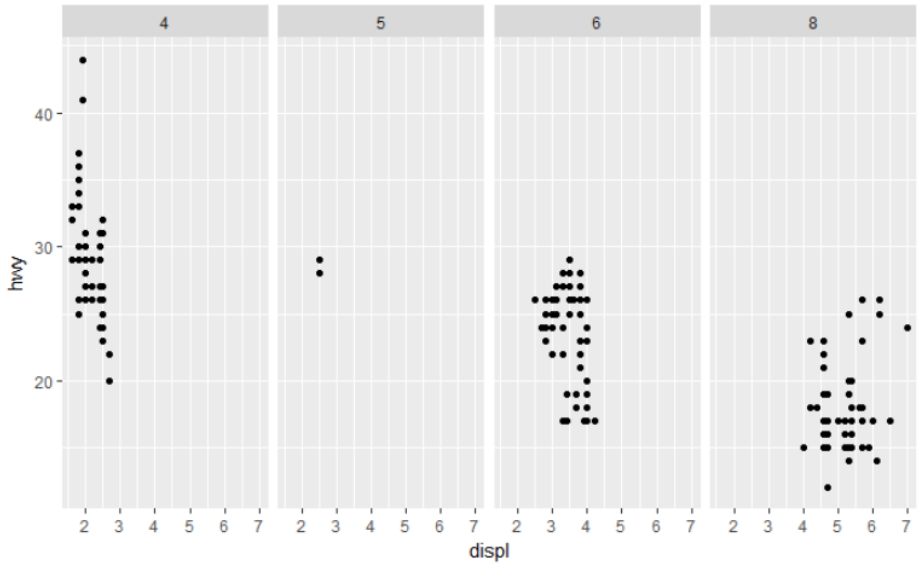
```
ggplot(data=mpg) +
  geom_point(mapping=aes(x=drv, y=cyl))
```

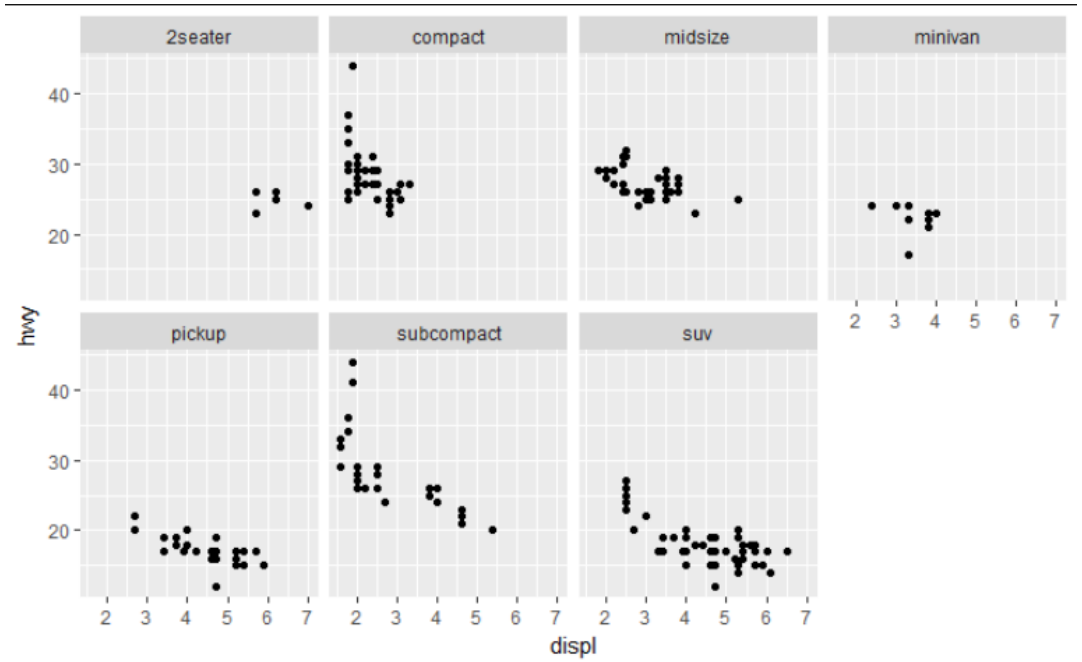
- ¿Qué sucede con nuestro *facet* si digitamos? ¿ Cuáles son sus diferencias?

```
ggplot(data=mpg) +
  geom_point(mapping = aes(x =  displ, y=hwy)) +
  facet_grid(drv ~.)

ggplot(data=mpg) +
  geom_point(mapping=aes(x=displ, y=hwy)) +
  facet_grid(. ~ cyl)
```

EJERCICIO 2. FACETS

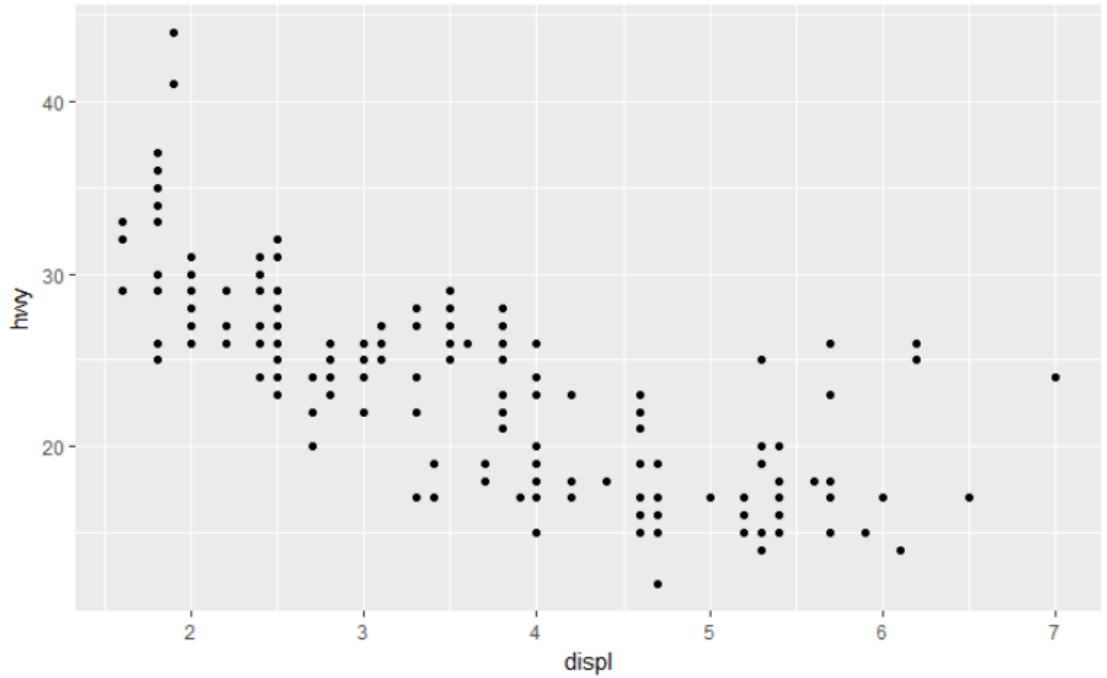
- ¿Cuáles son las ventajas y las desventajas de usar *facets*?
- Observe las propiedades del *facet_grid*



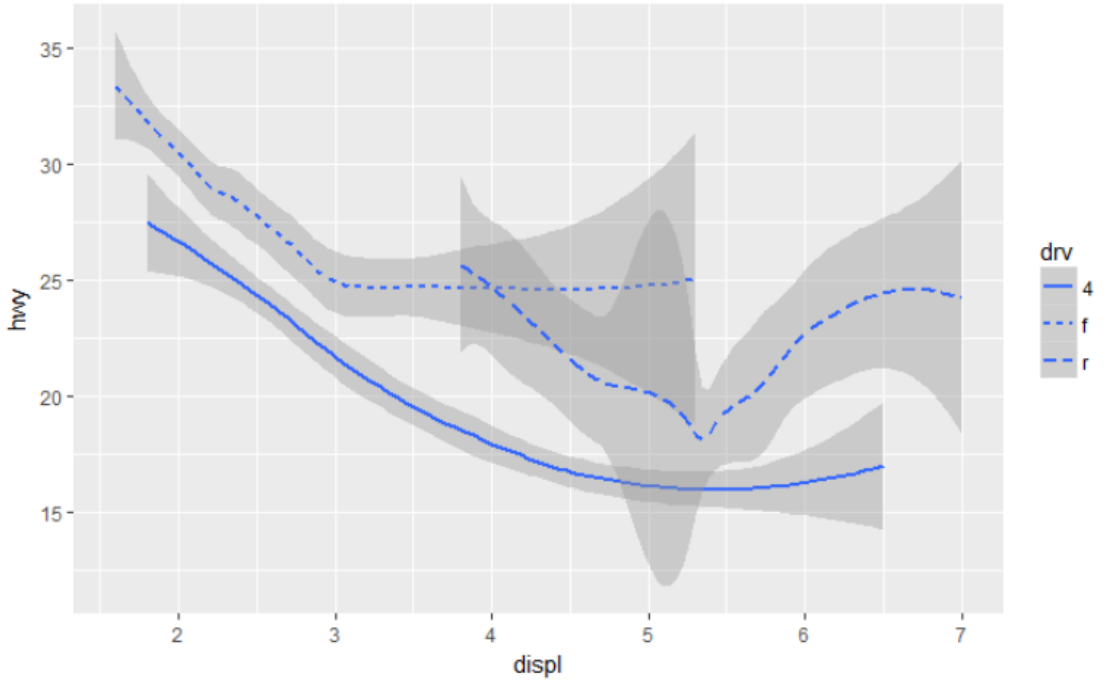


OBJETOS GEOMÉTRICOS

```
ggplot(data=mpg) +
  geom_point(mapping = aes(x =displ, y=hwy))
```




```
ggplot(data=mpg) +
  geom_smooth(mapping = aes(x =displ, y=hwy, linetype=drv))
```



OBJETOS GEOMÉTRICOS

- Cada objeto geométrico generado por las funciones de R (por ejemplo, geom_point y geom_smooth) tiene sus propias propiedades y así mismo la aplicación de *Aesthethic.*

Aesthetics

`geom_point` understands the following aesthetics (required aesthetics are in bold):

- **x**
- **y**
- alpha
- colour
- fill
- group
- shape
- size
- stroke

Aesthetics

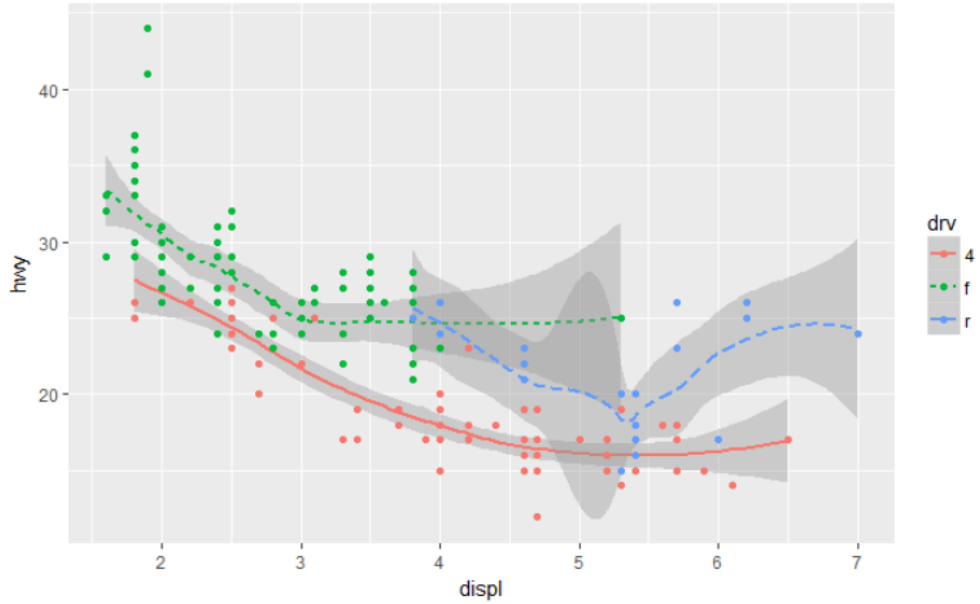
`geom_smooth` understands the following aesthetics (required aesthetics are in bold):

- **x**
- **y**
- alpha
- colour
- fill
- group
- linetype
- size
- weight

OBJETOS GEOMÉTRICOS

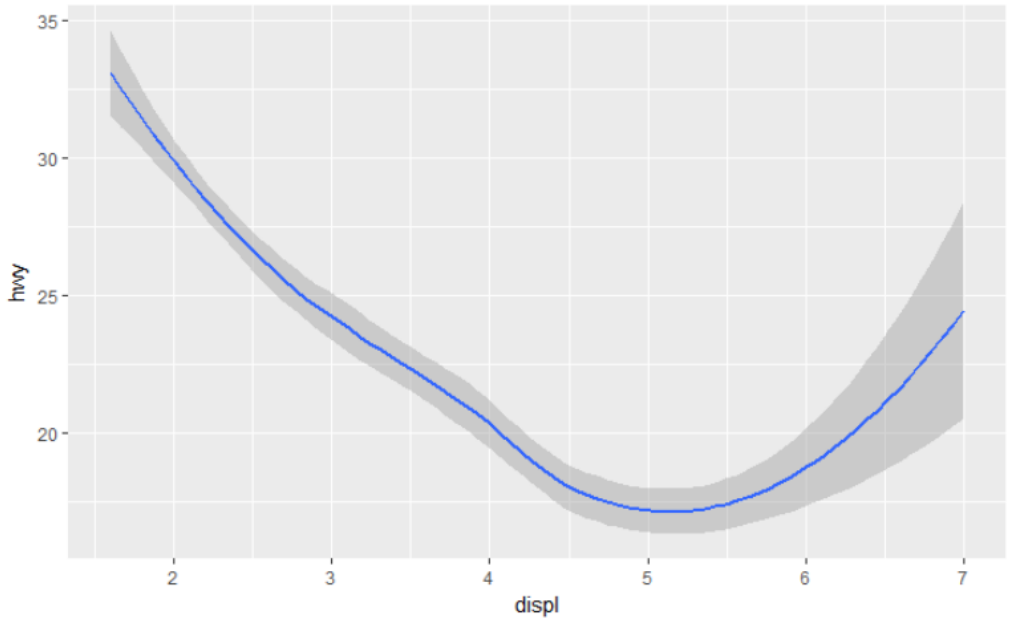
- Cada objeto geométrico generado por las funciones de R (por ejemplo, geom_point y geom_smooth) tiene sus propias propiedades y así mismo la aplicación de *Aesthethic.*

```
ggplot(data=mpg) +
  geom_smooth(mapping = aes(x =displ, y=hwy, linetype=drv, colour=drv))+
  geom_point(mapping = aes(x =displ, y=hwy, colour=drv))
```

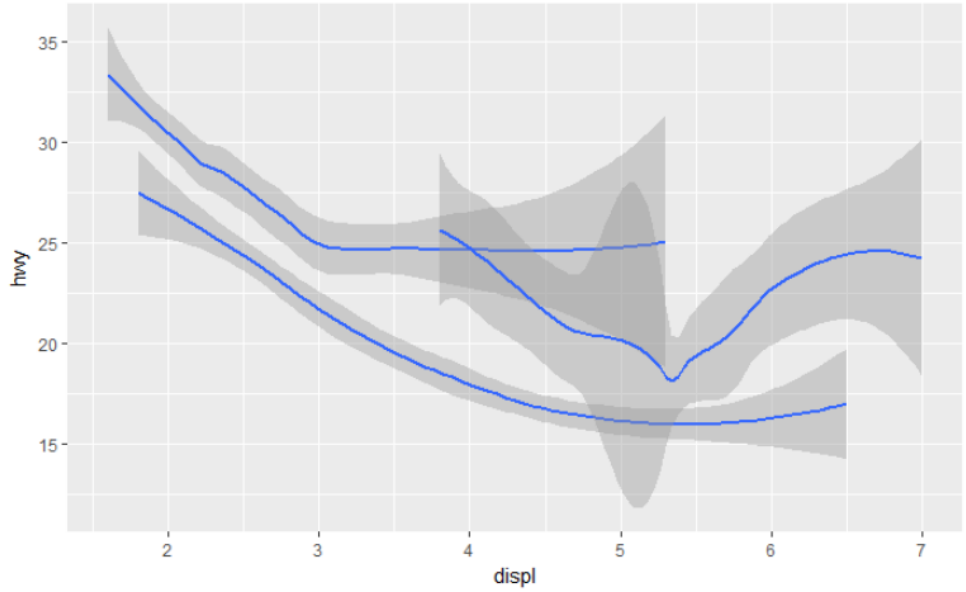


OBJETOS GEOMÉTRICOS

```
ggplot(data = mpg) +
  geom_smooth(mapping=aes(x=displ, y=hwy))
```

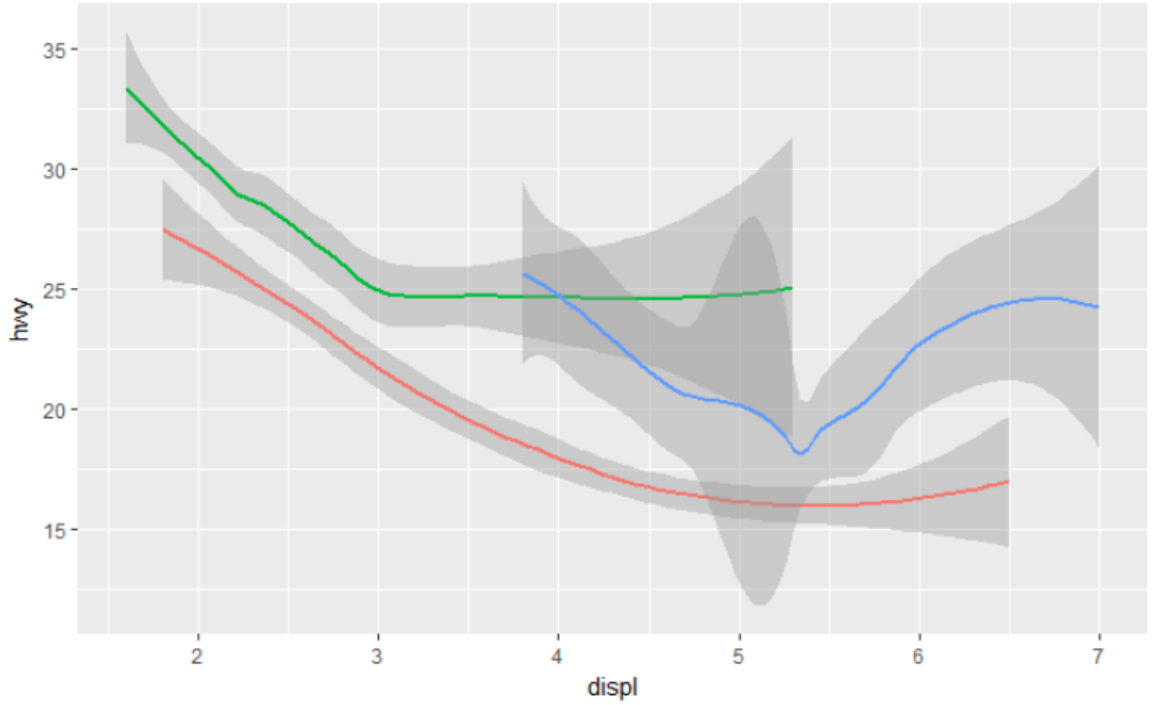



```
ggplot(data = mpg) +
  geom_smooth(mapping=aes(x=displ, y=hwy, group=drv))
```

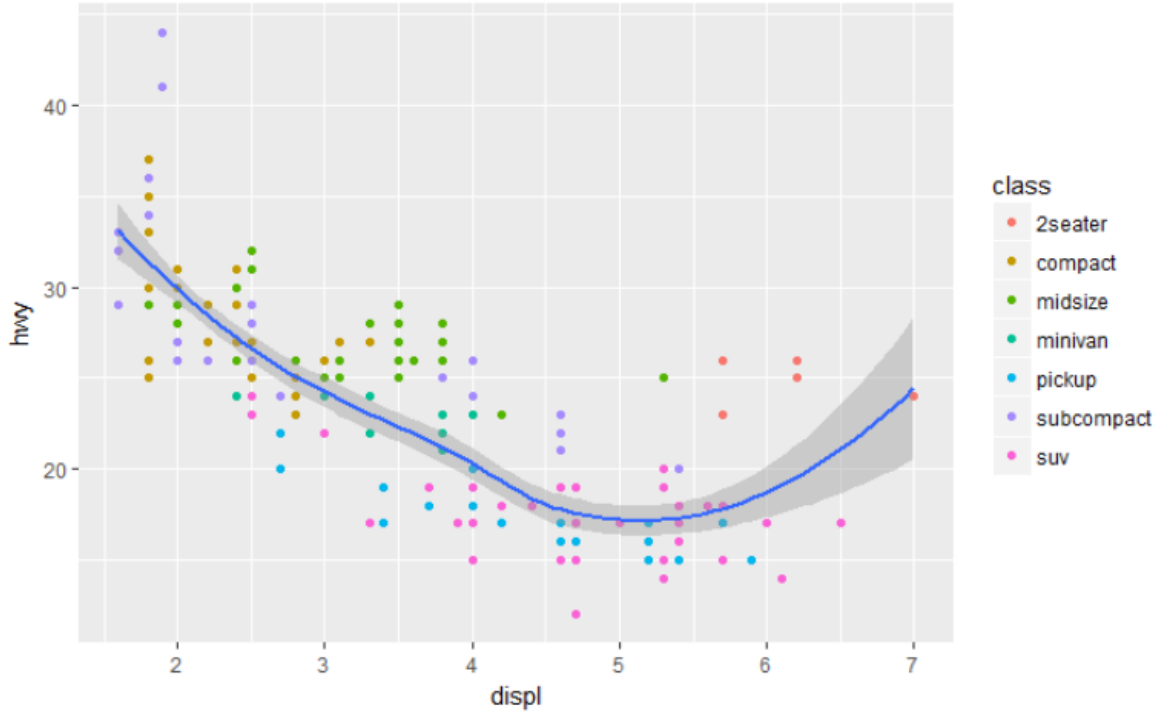


OBJETOS GEOMÉTRICOS

```
ggplot(data = mpg) +
  geom_smooth(mapping=aes(x=displ, y=hwy, group=drv, color=drv),
              show.legend = FALSE)
```

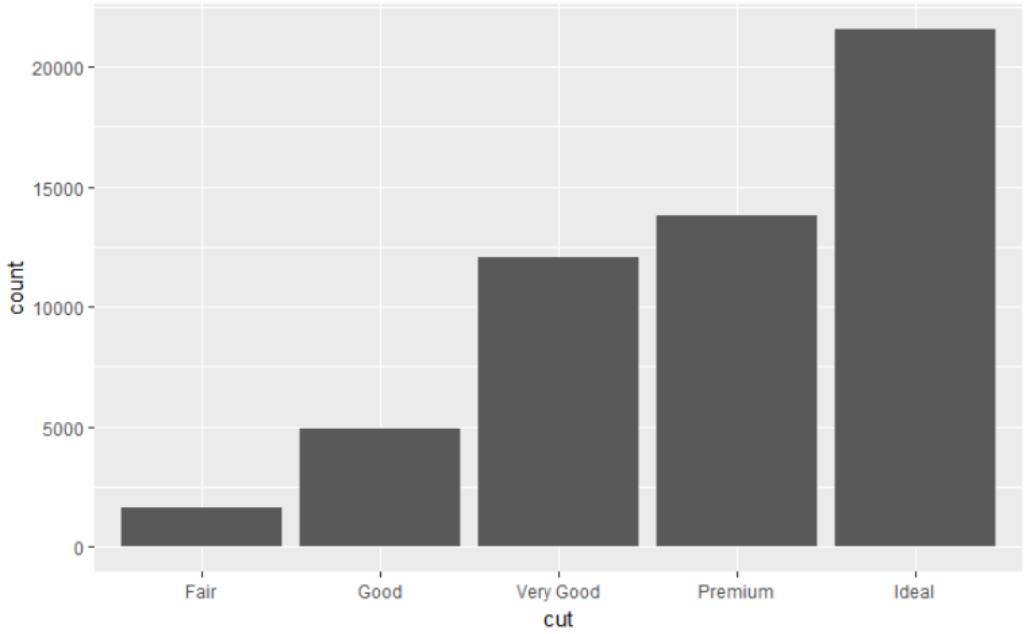



```
ggplot(data=mpg, mapping=aes(x =displ, y=hwy)) +
  geom_point(mapping = aes(colour=class)) +
  geom_smooth()
```



TIPOS DE GRÁFICOS

- **Gráfico de barras (bar chart)**, es un elemento que generalmente es usado para ver los valores de variables discretas, sus valores pueden ser ordinales o nominales, además, puede ser utilizada para ver la distribución discreta.

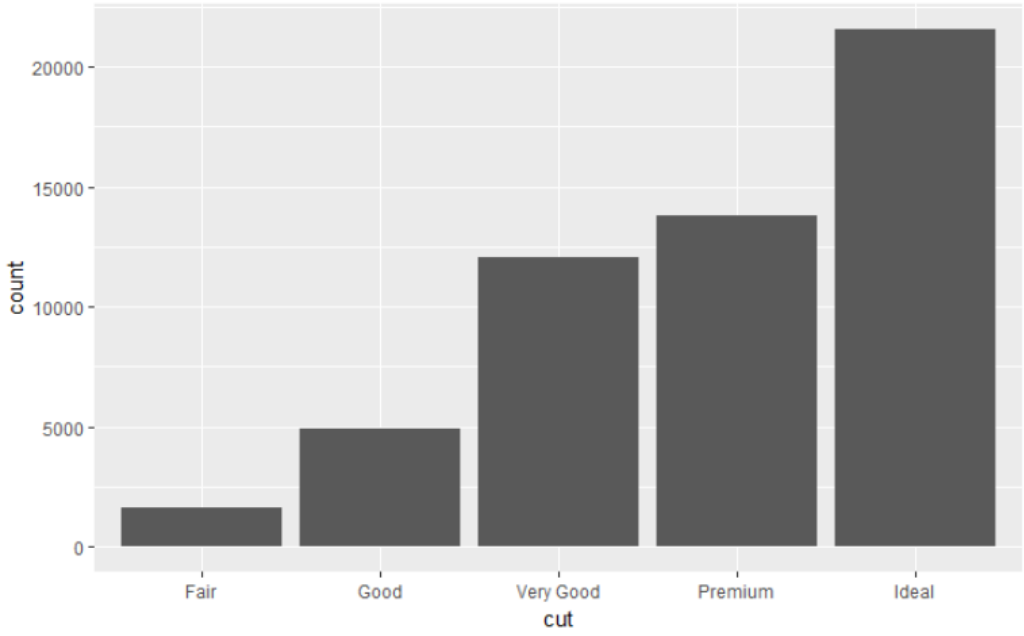


TIPOS DE GRÁFICOS

- Veamos el *dataset Diamonds*, este conjunto de datos se encuentra en la librería ggplot2, exploremos sus valores y tipos.

```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x=cut))
```

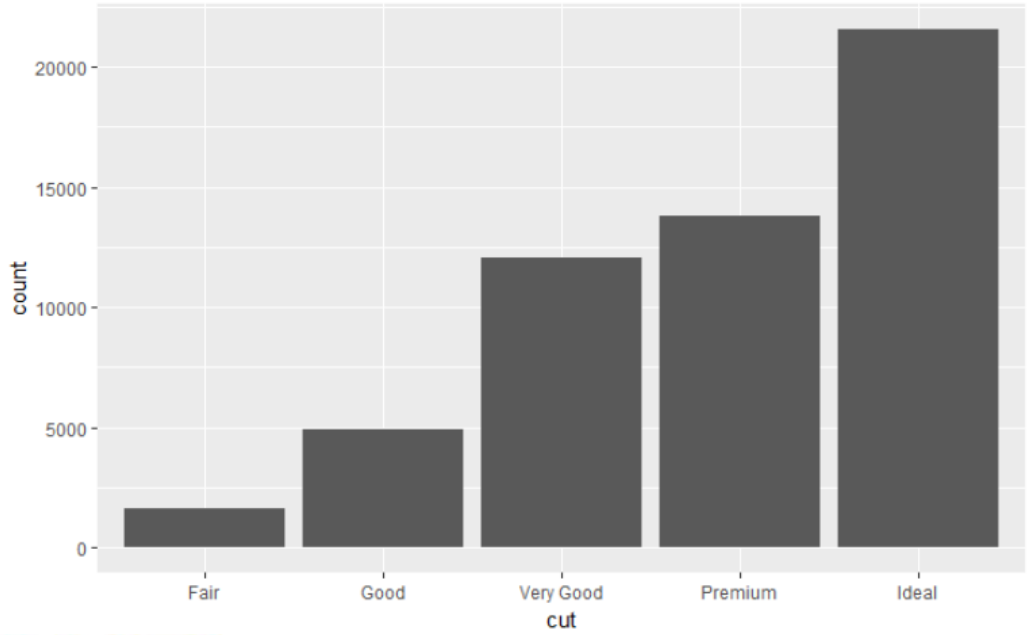
¿Encuentra algo particular en la generación de este gráfico?



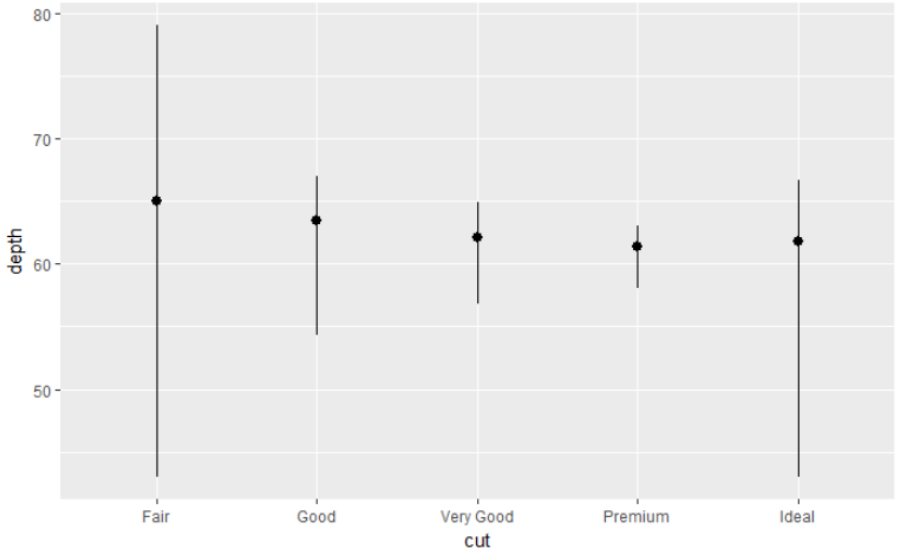
TIPOS DE GRÁFICOS

- Exploremos las propiedades de geom_bar

```
ggplot(data = diamonds) +
  stat_count(mapping = aes(x=cut))
```



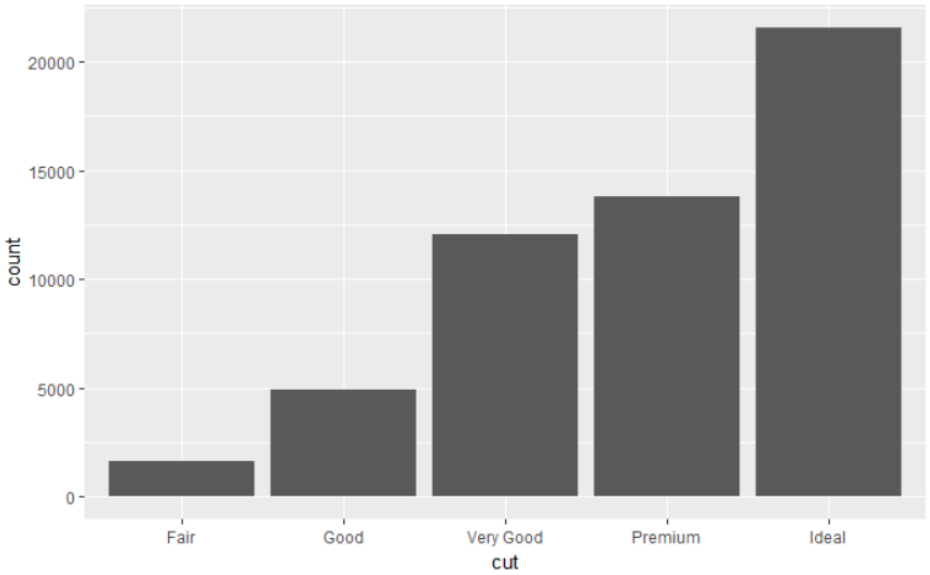

```
ggplot(data=diamonds) +
  stat_summary(
    mapping = aes(x=cut, y=depth),
    fun.ymin = min,
    fun.ymax = max,
    fun.y = median
  )
```



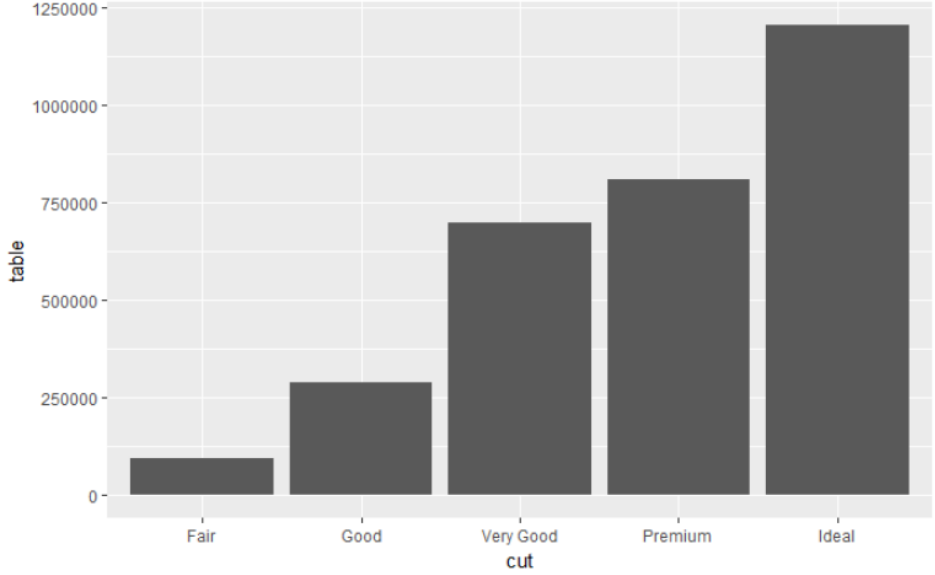
EJERCICIO 3. BAR CHART

- ¿Cuál es la diferencia entre las funciones **geom_bar** y **geom_col**?

```
ggplot(data=diamonds) +
  geom_bar(mapping = aes(x=cut))
```




```
ggplot(data=diamonds) +
  geom_col(mapping = aes(cut,table))
```



TIPOS DE GRÁFICOS

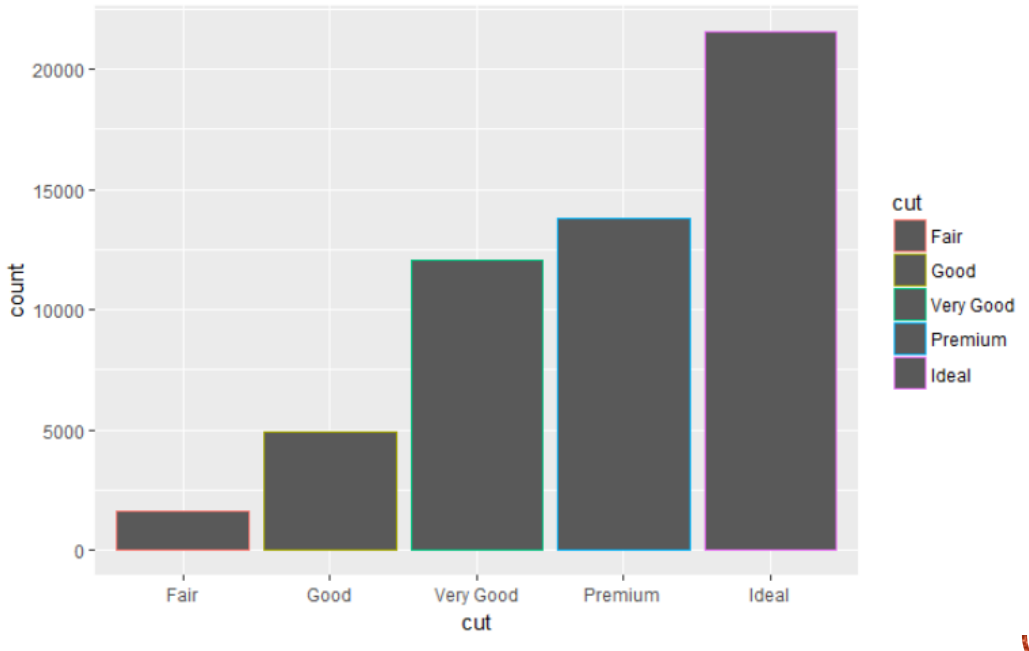
- Recordemos *Aesthetic.*

Aesthetics

`geom_bar` understands the following aesthetics (required aesthetics are in bold):

- **`x`**
- **`y`**
- `alpha`
- `colour`
- `fill`
- `group`
- `linetype`
- `size`

```
ggplot(data=diamonds) +
  geom_bar(mapping=aes(x=cut, colour=cut))
```



TIPOS DE GRÁFICOS

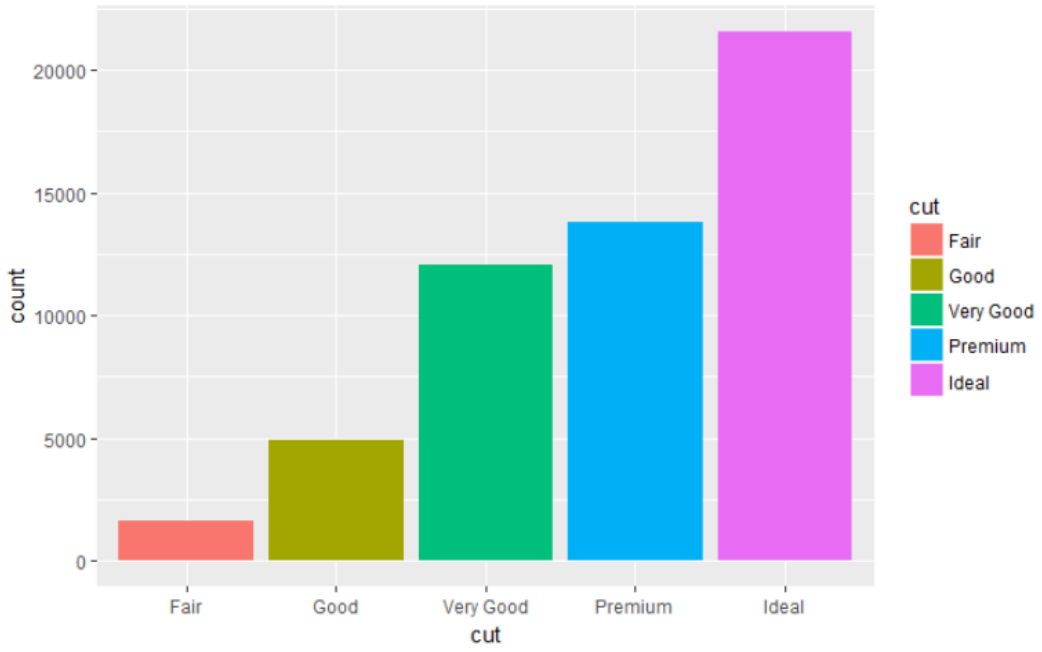
- Recordemos *Aesthetic.*

Aesthetics

`geom_bar` understands the following aesthetics (required aesthetics are in bold):

- **`x`**
- **`y`**
- `alpha`
- `colour`
- `fill`
- `group`
- `linetype`
- `size`

```
ggplot(data=diamonds) +
  geom_bar(mapping=aes(x=cut, fill=cut))
```



TIPOS DE GRÁFICOS

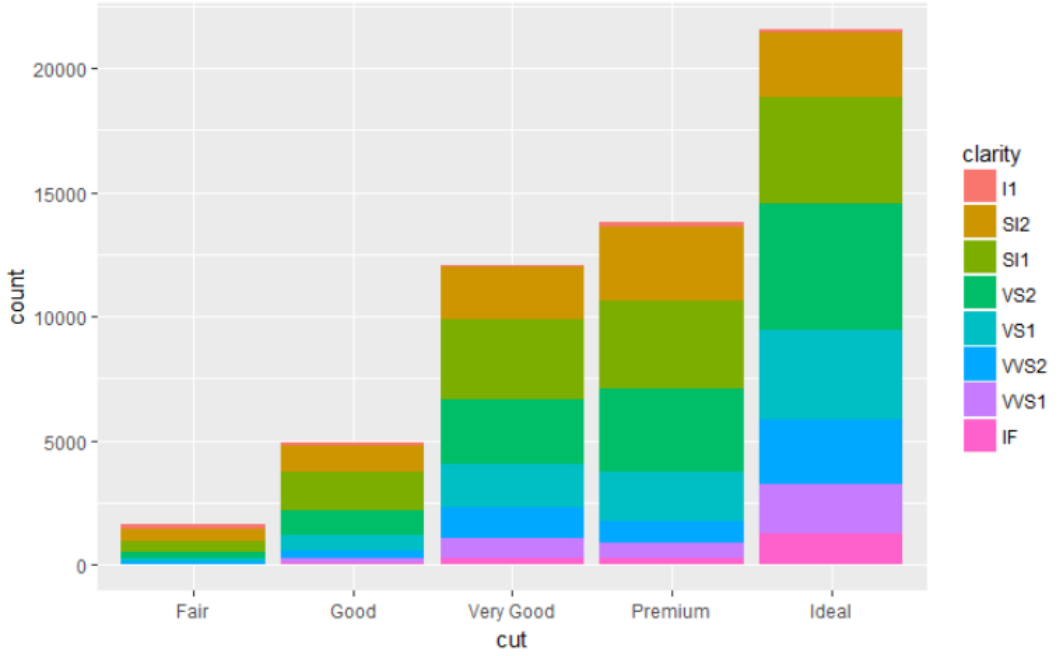
- Recordemos *Aesthetic.*

Aesthetics

`geom_bar` understands the following aesthetics (required aesthetics are in bold):

- **`x`**
- **`y`**
- `alpha`
- `colour`
- `fill`
- `group`
- `linetype`
- `size`

```
ggplot(data=diamonds) +
  geom_bar(mapping=aes(x=cut, fill=clarity))
```



TIPOS DE GRÁFICOS

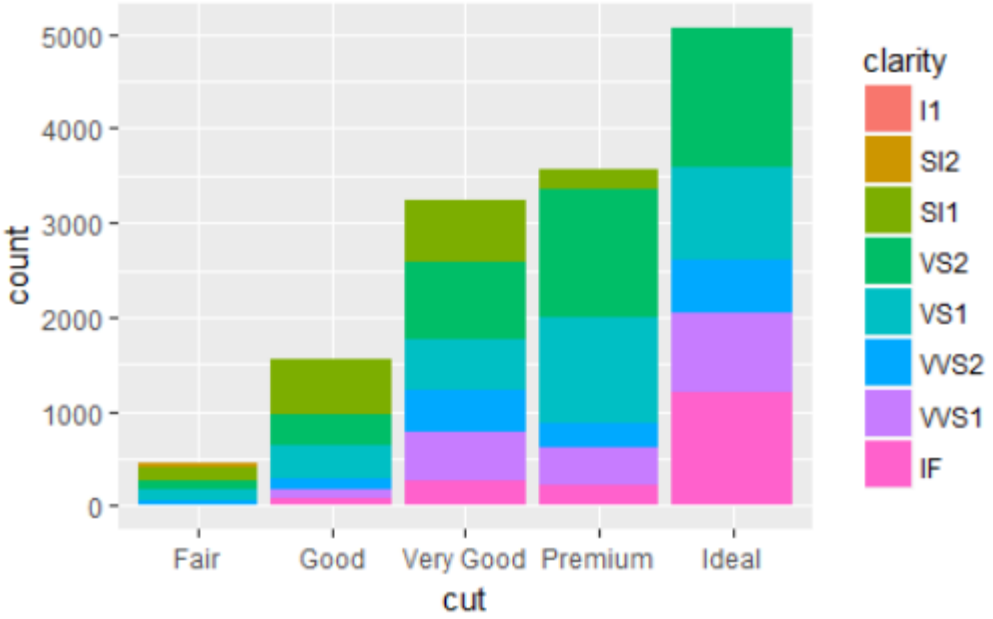
- Recordemos *Aesthetic.*

Aesthetics

`geom_bar` understands the following aesthetics (required aesthetics are in bold):

- **x**
- **y**
- `alpha`
- `colour`
- `fill`
- `group`
- `linetype`
- `size`

```
ggplot(data=diamonds) +
  geom_bar(mapping=aes(x=cut, fill=clarity),
position="identity")
```



TIPOS DE GRÁFICOS

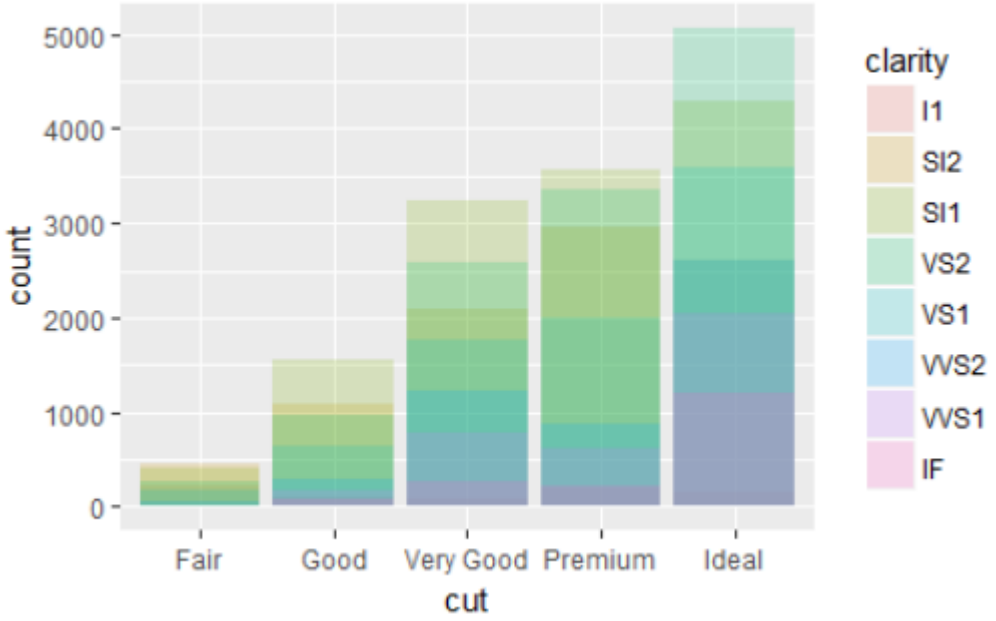
- Recordemos *Aesthetic.*

Aesthetics

`geom_bar` understands the following aesthetics (required aesthetics are in bold):

- **x**
- **y**
- alpha
- colour
- fill
- group
- linetype
- size

```
ggplot(data=diamonds,
       mapping=aes(x=cut, fill=clarity)) +
  geom_bar(alpha=1/5, position="identity")
```



TIPOS DE GRÁFICOS

- **Gráfico de líneas (Line Graph),** es un elemento gráfico que permite visualizar datos numéricos ordinales. A diferencia del gráfico de barras, este insumo conecta varios puntos a través de una línea. Usualmente este gráfico es utilizado para analizar datos a través del tiempo o tendencias. No debería emplearse para ver datos nominales.

EJERCICIO 4. BICLETAS

- ¡Exploremos otras fuentes de información!
- Cree un nuevo data.frame del archivo day.csv obtenido del repositorio de *Bike Sharing Dataset Data Set*

```
daily_data = read.csv(file.choose(), header=TRUE, stringsAsFactors = FALSE)
```

- ¿Cuáles son las variables y de que tipo son?
- Cambie el tipo de la variable *dteday a* Date.

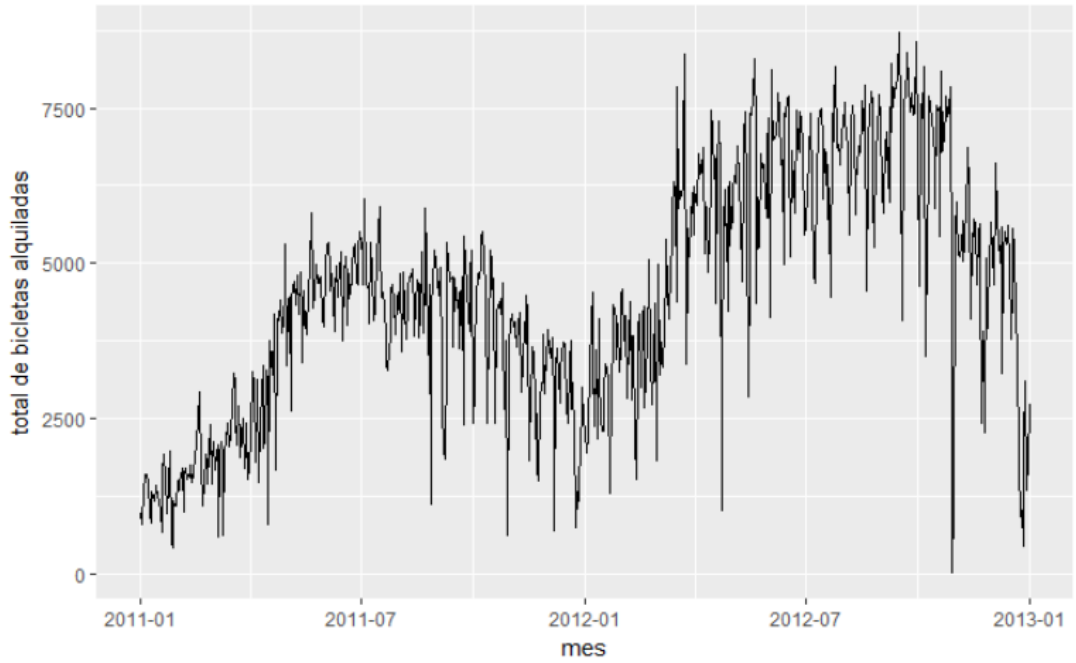
```
daily_data$Date = as.Date(daily_data$dteday)
```

- ¿Cómo ha sido el comportamiento del alquiler de las bicicletas durante los años 2011-2013?

EJERCICIO 4. BICLETAS

- Creemos un gráfico de líneas; recuerde que este insumo sirve para visualizar datos numéricos ordinales.
- ¿Qué variables ordinales existen en nuestro date.frame?

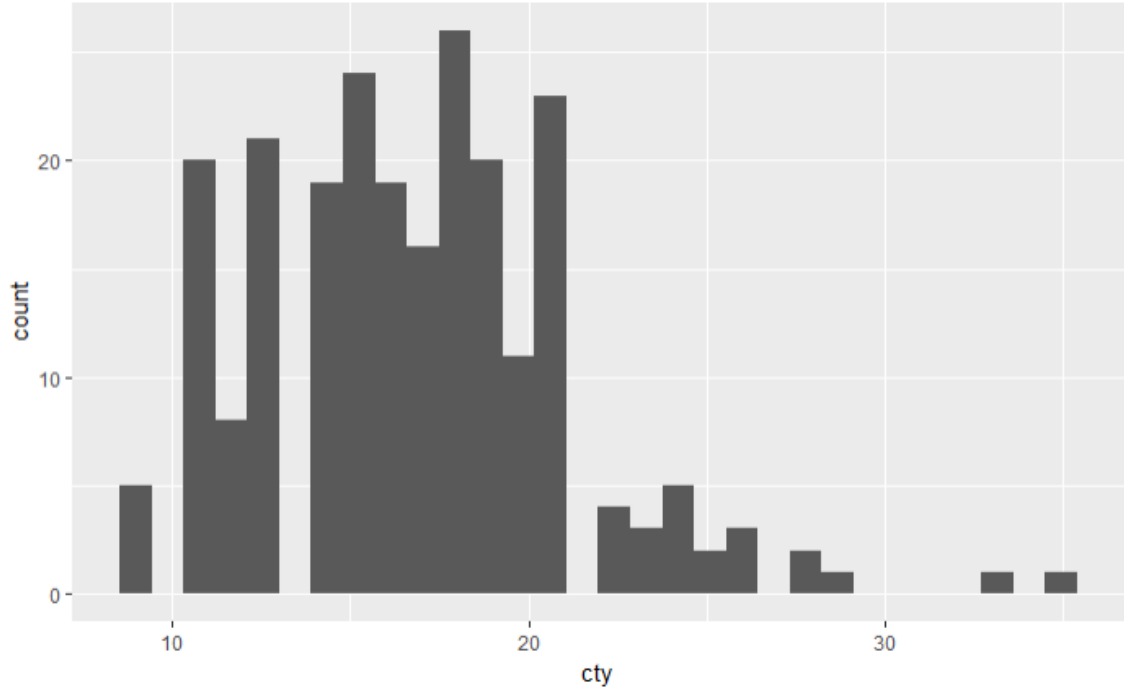
```
ggplot(daily_data, aes(Date, cnt)) +
  geom_line() +
  scale_x_date("month") +
  ylab("total de bicletas alquiladas") +
  xlab("mes")
```



TIPOS DE GRÁFICOS

- **Histograma,** es un tipo de gráfico de barras utilizado para variables continuas que se encuentran agrupadas en intervalos. Adicionalmente, este insumo permite en ocasiones observar la distribución de los datos y cada área de las barras es proporcional a sus frecuencias.

```
ggplot(data = mpg, aes(cty)) +
  geom_histogram()
```

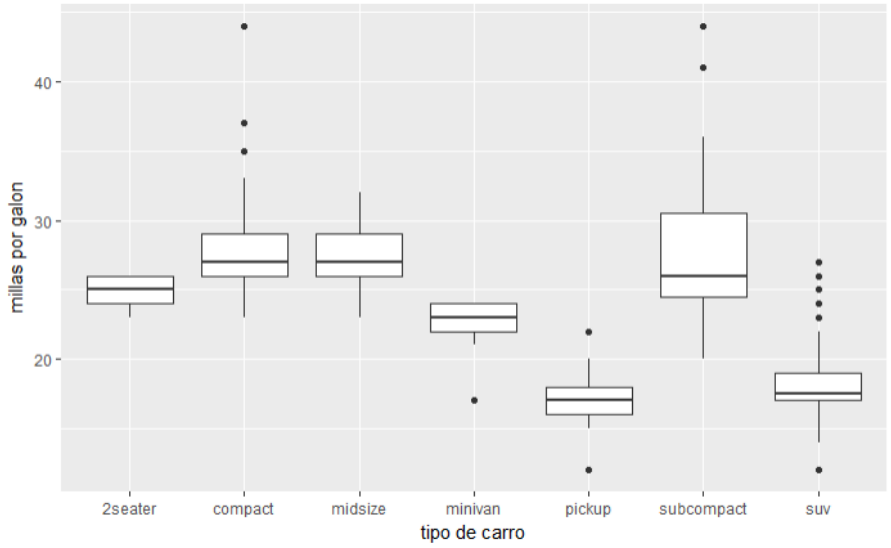


TIPOS DE GRÁFICOS

- **Caja de bigotes (Box Plot)**, es un insumo gráfico que permite visualizar la mediana, el rango, Q1, Q3 y el rango intercuartílico (IQR) de una distribución.
- Es ideal para comparar múltiples distribuciones.
- Permite la identificación de valores atípicos.
- Una caja posicionada en el centro de los bigotes representa una distribución normal.

```
ggplot(data = mpg, aes(class, hwy)) +
  geom_boxplot() +
  xlab("tipo de carro") +
  ylab("millas por galon")
```

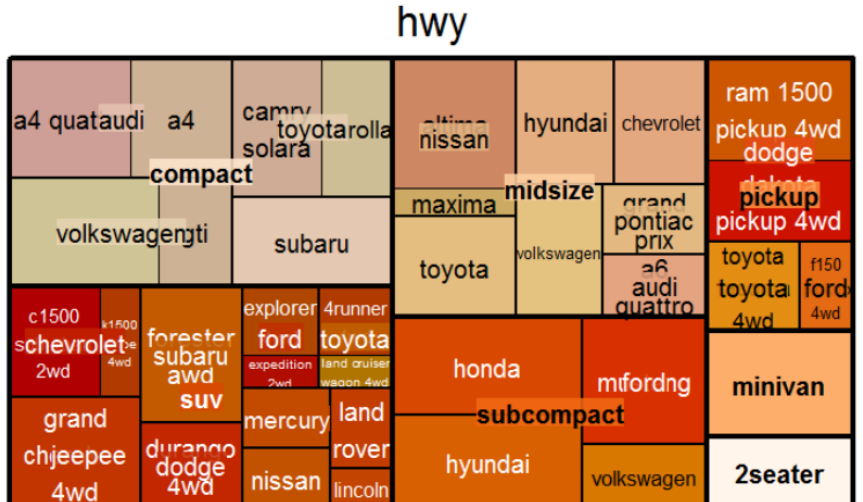
```
ggplot(data = mpg, aes(class, hwy)) +
  geom_boxplot() +
  xlab("tipo de carro") +
  ylab("millas por galon") +
  coord_flip()
```



TIPOS DE GRÁFICOS

- **Treemap**, es un insumo gráfico que permite visualizar datos jerárquicos (estructura de árbol) como un conjunto de rectángulos anidados. Cada rama del árbol se le da un rectángulo, que es posteriormente enlazado con rectángulos más pequeños que representan sub-ramas. El rectángulo de un nodo hoja tiene una superficie proporcional a una dimensión especificada de los datos.

  Cuando las dimensiones de color y tamaño están correlacionados de alguna manera con la estructura del árbol, a menudo se puede ver fácilmente los patrones entre los datos.



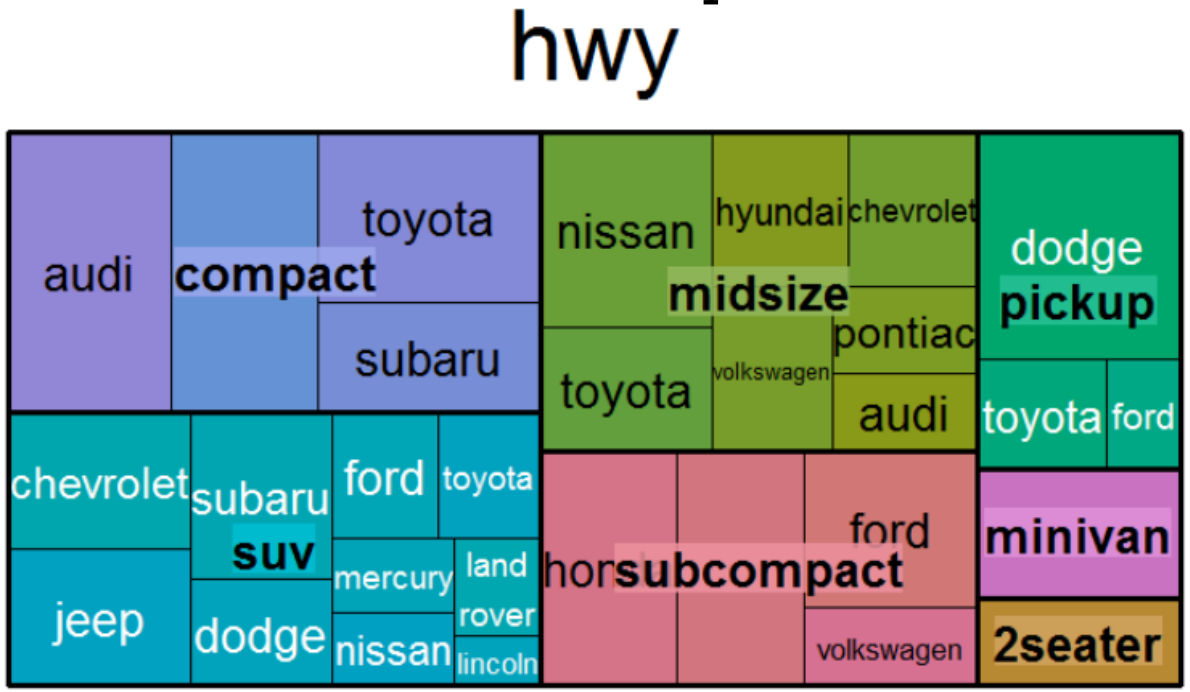
EJERCICIO 5. TREEMAP

- Revisemos nuevamente nuestro *dataset* de carros ***mpg***
- Es importante que antes de aplicar el *treemap* no existan registros incompletos ya que podrían surgir excepciones durante el proceso de agrupación.
- Creemos un nuevo data.frame con los registros del año 2008.
- Creemos un treemap que nos permita identificar grupos primarios de la clase de vehículo y el fabricante.

```
library(treemap)

summary(mpg)
mpg2008 <- subset(mpg, mpg$year == 2008)

head(mpg2008, 3)

treemap <- treemap(mpg2008, c("class","manufacturer"), vSize="hwy", vColor="cty",
          fontsize.labels = 20, fontsize.title = 42)
```

EJERCICIO 5. TREEMAP

- El área de cada baldosa corresponde al consumo en autopista promedio de los vehículos que ese fabricante tiene de esa clase.
- El tono de la baldosa hace referencia a las clases de vehículos.
- Las variaciones de claridad o brillo corresponden al consumo promedio en la ciudad.

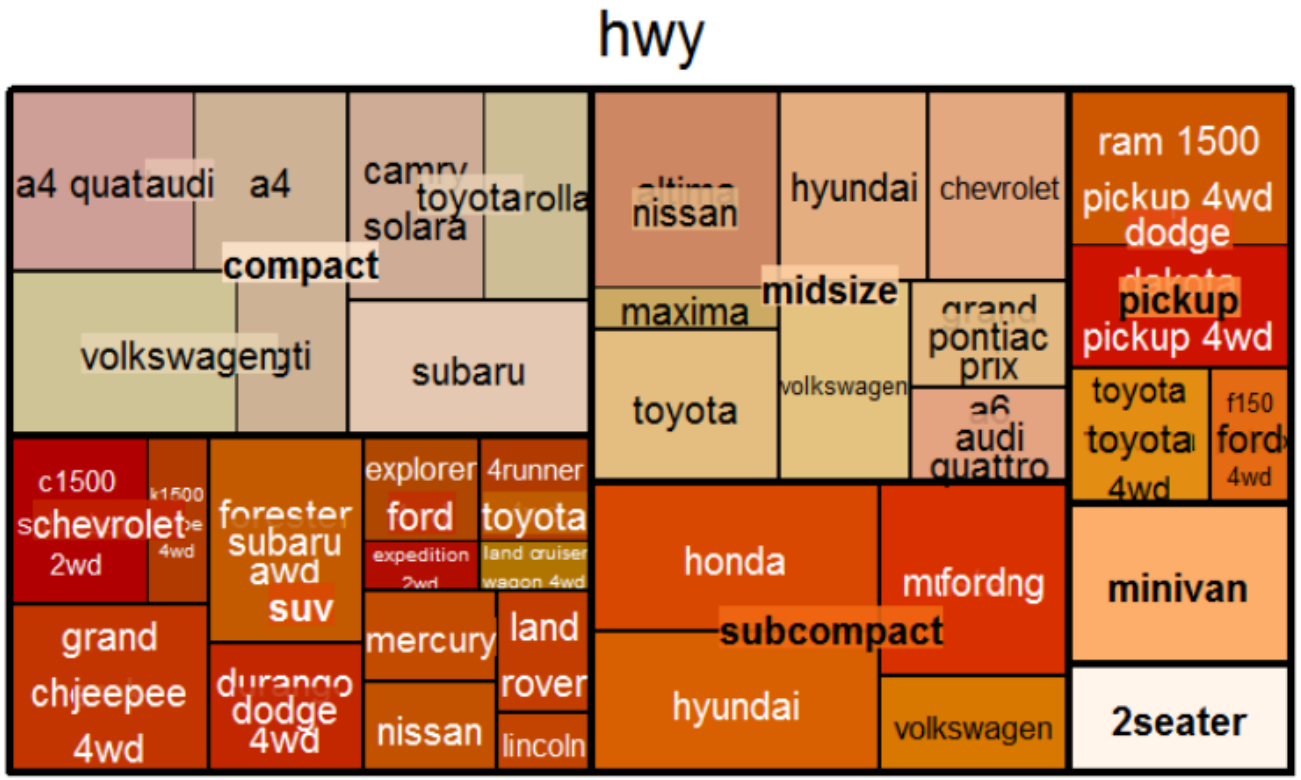


EJERCICIO 5. TREEMAP

- Sin embargo comparar las dos variables de consumo en autopista y en la ciudad con atributos diferentes no es correcto, porque la percepción visual de magnitudes a través de áreas no es equiparable con la percepción visual de magnitudes a través del color. A continuación, presentamos otra representación gráfica a través de una paleta de colores.
- ?RColorBrewer

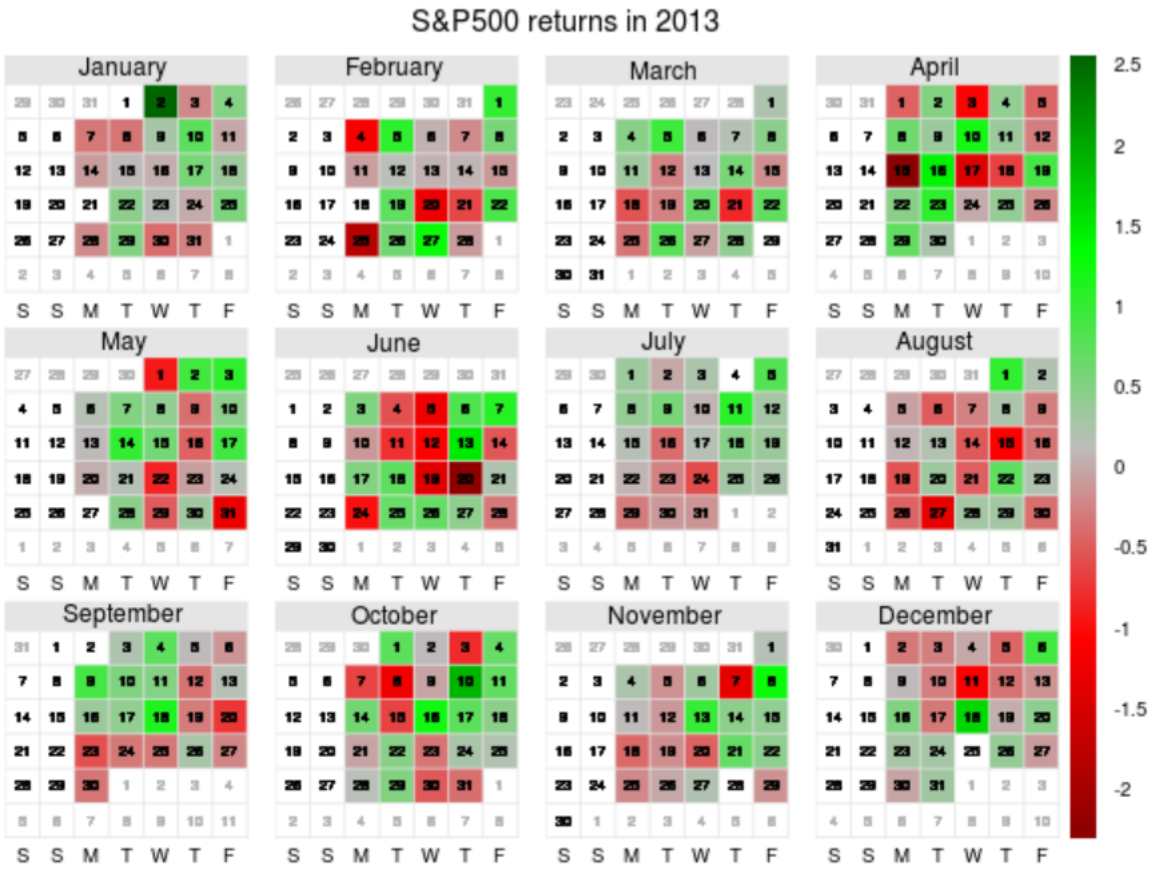
EJERCICIO 5. TREEMAP

```
treemapHwy <- treemap(mpg2008, c("class", "manufacturer", "model"), vSize= "hwy", vColor="displ", palette ="Oranges",
fontsize.labels = 15, fontsize.title = 25)
```



CALENDAR PLOTS

- Aplicación especial de los heatmaps.
- Los heatmaps o mapas de calor son representaciones gráficas de datos donde los valores se representan como colores.
- Sirven para:
  - proporcionar un resumen visual de la información.
  - Comprender conjuntos de datos complejos.



http://rbagd.eu/2014-03-25/financials-with-openair.html

CALENDAR PLOTS

```
library(openair)

calendarPlot(mydata, pollutant = "nox", year = 2003, month = 1:12, type = "default", annotate = "date",
statistic = "mean", cols = "heat", limits = c(0, 100), lim = NULL, col.lim = c("grey30", "black"),
 font.lim = c(1, 2), cex.lim = c(0.6, 1), digits = 0, data.thresh = 0, labels = NA, breaks = NA, w.shift = 0,
main = paste(pollutant, "in", year), key.header = "", key.footer = "", key.position = "right", key = TRUE,
 auto.text = TRUE, ...)
```

- Mydata: A data frame minimally containing date and at least one other numeric variable. The date should be in either Date format or class POSIXct.
- Pollutant: Mandatory. A pollutant name corresponding to a variable in a data frame should be supplied e.g. pollutant = "nox".
- Year:Year to plot e.g. year = 2003.
- Month:If only certain month are required. By default the function will plot an entire year even if months are missing. To only plot certain months use the month option where month is a numeric 1:12 e.g. month = c(1, 12) to only plot January and December.
- Statistic: Statistic passed to timeAverage.
- Cols: Colours to be used for plotting. Options include “default”, “increment”, “heat”, “jet”.

CALENDAR PLOTS

Los Calendar Plots frecuentemente se aplican a series financieras. Para eso vamos a instalar y activar la librería quantmod y después:

```
> getSymbols("YHOO",src="google")
[1] "YHOO"
> data <- as.data.frame(YHOO)
> data$returns <- with(data, (YHOO.Close/YHOO.Open-1)*100)
> data$date <- as.Date(rownames(data))
>
> calendarPlot(data, pollutant="returns", year=2017,
+              cols=c("darkred", "red", "gray", "green", "darkgreen"),
+              main="Retornos yahoo % (2018)")
>
> calendarPlot(data, pollutant="returns", year=2018,
+              cols=c("darkred", "red", "gray", "green", "darkgreen"),
+              main="Retornos yahoo % (2018)")
>
> calendarPlot(data, pollutant="returns", year=2018,
+              cols="heat",
+              main="Retornos yahoo % (2018)")
>
```

CALENDAR PLOTS

- Pero los Calendar Plots también se pueden aplicar a otros escenarios:

```
> datos<-read_xlsx("Online Retail.xlsx")
> datos<-as.data.frame(datos)
> head(datos)
  InvoiceNo StockCode                         Description Quantity
1    536365    85123A  WHITE HANGING HEART T-LIGHT HOLDER        6
2    536365     71053                 WHITE METAL LANTERN        6
3    536365    84406B      CREAM CUPID HEARTS COAT HANGER        8
4    536365    84029G KNITTED UNION FLAG HOT WATER BOTTLE        6
5    536365    84029E      RED WOOLLY HOTTIE WHITE HEART.        6
6    536365     22752        SET 7 BABUSHKA NESTING BOXES        2
          InvoiceDate UnitPrice CustomerID        Country
1 2010-12-01 08:26:00      2.55      17850 United Kingdom
2 2010-12-01 08:26:00      3.39      17850 United Kingdom
3 2010-12-01 08:26:00      2.75      17850 United Kingdom
4 2010-12-01 08:26:00      3.39      17850 United Kingdom
5 2010-12-01 08:26:00      3.39      17850 United Kingdom
6 2010-12-01 08:26:00      7.65      17850 United Kingdom
```

Daqing Chen, Sai Liang Sain, and Kun Guo, Data mining for the online retail industry: A case study of RFM model-based customer segmentation using data mining, Journal of Database Marketing and Customer Strategy Management, Vol. 19, No. 3, pp. 197â€“208, 2012 (Published online before print: 27 August 2012. doi: 10.1057/dbm.2012.17)

CALENDAR PLOTS

¿De las visualizaciones que generó que podría concluir?

```
> datos<-subset(datos, Country=="Germany")
>
> extractdate <- function(date) {
+   day <- format(date, format="%d")
+   month <- format(date, format="%m")
+   year <- format(date, format="%Y")
+
+   cbind(day, month, year)
+ }
>
> datos<-cbind(datos,extractdate(datos$InvoiceDate))
> datos[,c(9:11)]<-sapply(datos[,c(9:11)],as.character)
>
> datosagg<-aggregate(Quantity ~ year+month+day, data= datos,sum, na.action=NULL)
> datosagg$fecha<-paste0(datosagg$year,datosagg$month,datosagg$day)
> datosagg$date<-as.Date(datosagg$fecha,format="%Y%m%d")
>
> calendarPlot(datosagg, pollutant="Quantity", year=2010,
+              cols="heat",
+              main="Envíos a Alemania (2010)")

> calendarPlot(datosagg, pollutant="Quantity", year=2011,
+              cols="heat",
+              main="Envíos a Alemania (2011)")
>
```