Abril de 2021

## Índice

1. La gramática de ggplot
2. Datos y mapeos estéticos
3. Capas: *Geoms*
4. Transformaciones estadísticas: *Stats*
5. División en subconjuntos: *Facetting*
6. Coordenadas y ejes: *Scales*
7. Temas
8. Leyendas
9. Múltiples gráficos

## 1. La gramática de ggplot: Componentes de un gráfico

* **Datos**.
* Mapeos estéticos (**aesthetic mappings**): vinculación entre variables presentes en los datos y propiedades visuales/ atributos estéticos.
* Una capa que describa como mostrar cada observación o agregación de observaciones (normalmente, mediante algún tipo de **objeto geométrico**). En definitiva, es *“lo que vemos”:* puntos, líneas, barras, etc.
* Transformaciones estadísticas.
* Escalas.
* Sistema de coordenadas.
* División de subconjuntos.

## 2. Carga de librerías y datos

#install.packages('ggplot2')  
library(ggplot2)  
library(dplyr)  
  
data(diamonds)  
str(diamonds)

## tibble [53,940 x 10] (S3: tbl\_df/tbl/data.frame)  
## $ carat : num [1:53940] 0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...  
## $ cut : Ord.factor w/ 5 levels "Fair"<"Good"<..: 5 4 2 4 2 3 3 3 1 3 ...  
## $ color : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 2 2 2 6 7 7 6 5 2 5 ...  
## $ clarity: Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<..: 2 3 5 4 2 6 7 3 4 5 ...  
## $ depth : num [1:53940] 61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...  
## $ table : num [1:53940] 55 61 65 58 58 57 57 55 61 61 ...  
## $ price : int [1:53940] 326 326 327 334 335 336 336 337 337 338 ...  
## $ x : num [1:53940] 3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...  
## $ y : num [1:53940] 3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...  
## $ z : num [1:53940] 2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...

*Documentación sobre el set de datos* [*aquí*](http://docs.ggplot2.org/0.9.3.1/diamonds.html)*.*

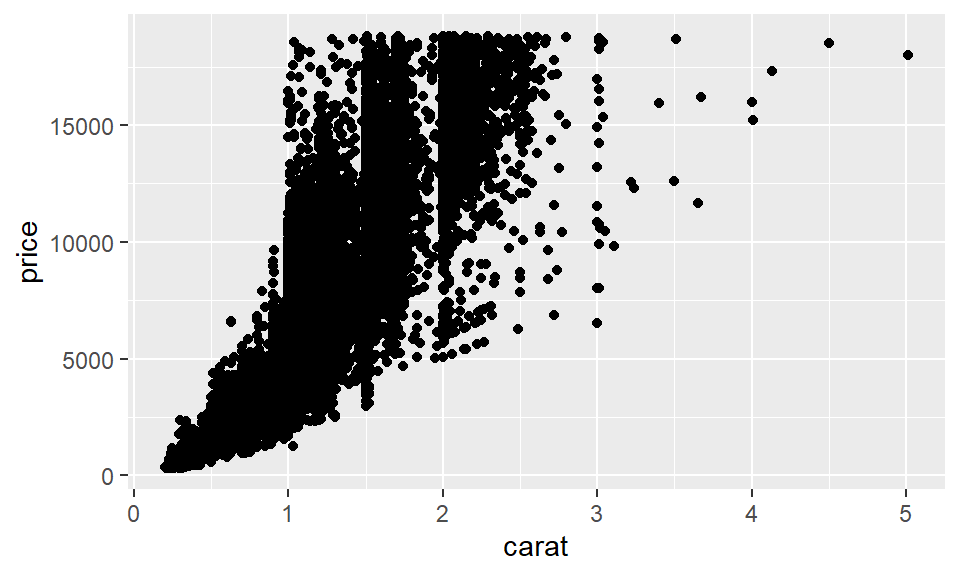
## Componentes obligatorios de un gráfico (I)

* **Datos**.
* Mapeos estéticos (**aesthetic mappings**) entre variables presentes en los datos y propiedades visuales.
* Una capa que describa como mostrar cada observación o agregación de observaciones (normalmente, mediante algún tipo de **objeto geométrico**).

## Un ejemplo (muy) sencillo

Queremos mostrar la relación entre peso y precio de diamantes.

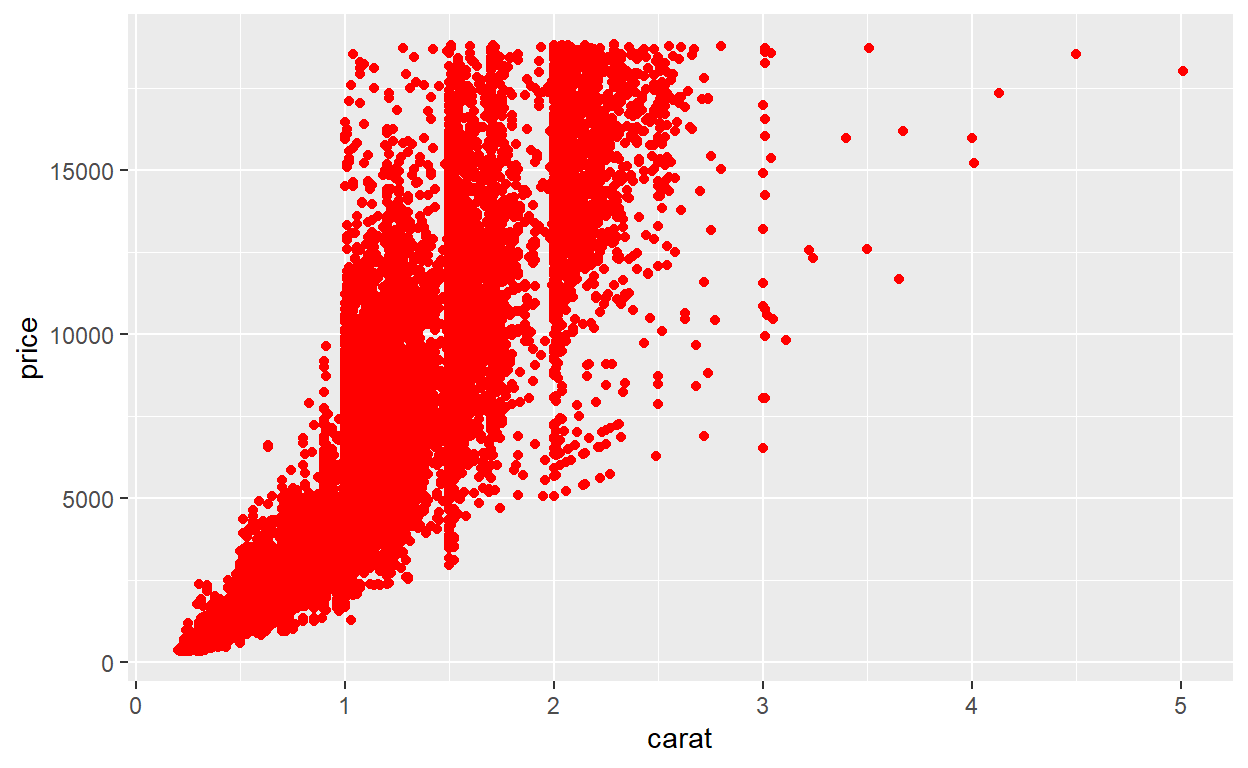
* data: el dataset diamonds.
* aesthetics: x = carat, y = price.
* geom: un punto para representar cada observación.

ggplot(data = diamonds, aes(x = carat, y = price)) +   
 geom\_point()

## Componentes obligatorios de un gráfico (II)

Sólo con los componentes obligatorios de un gráfico (datos, *aesthetics* y *geoms*) ya se pueden hacer bastantes cosas. Por ejemplo:

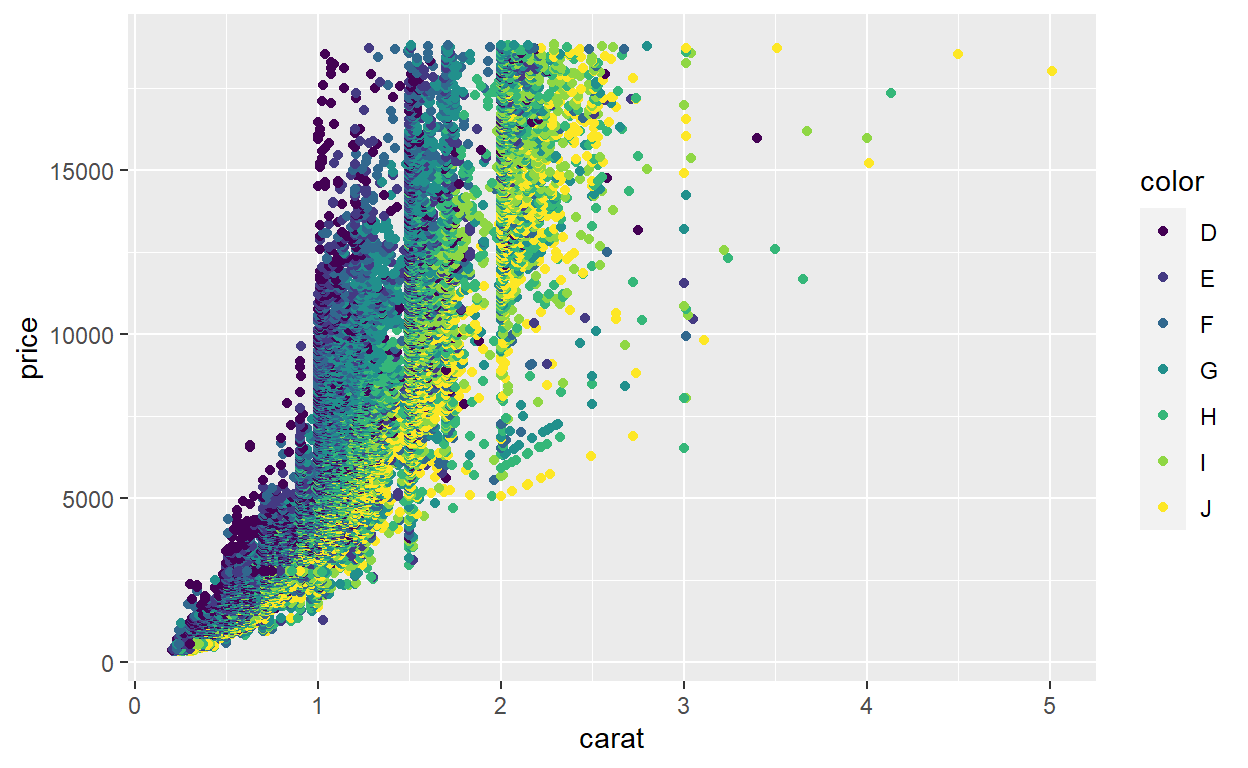
ggplot(data = diamonds, aes(x = carat, y = price)) +   
 geom\_point(col = 'red')



## Componentes obligatorios de un gráfico (III)

Aunque seguramente esto sea más interesante:

ggplot(data = diamonds, aes(x = carat, y = price, col = color)) +   
 geom\_point()



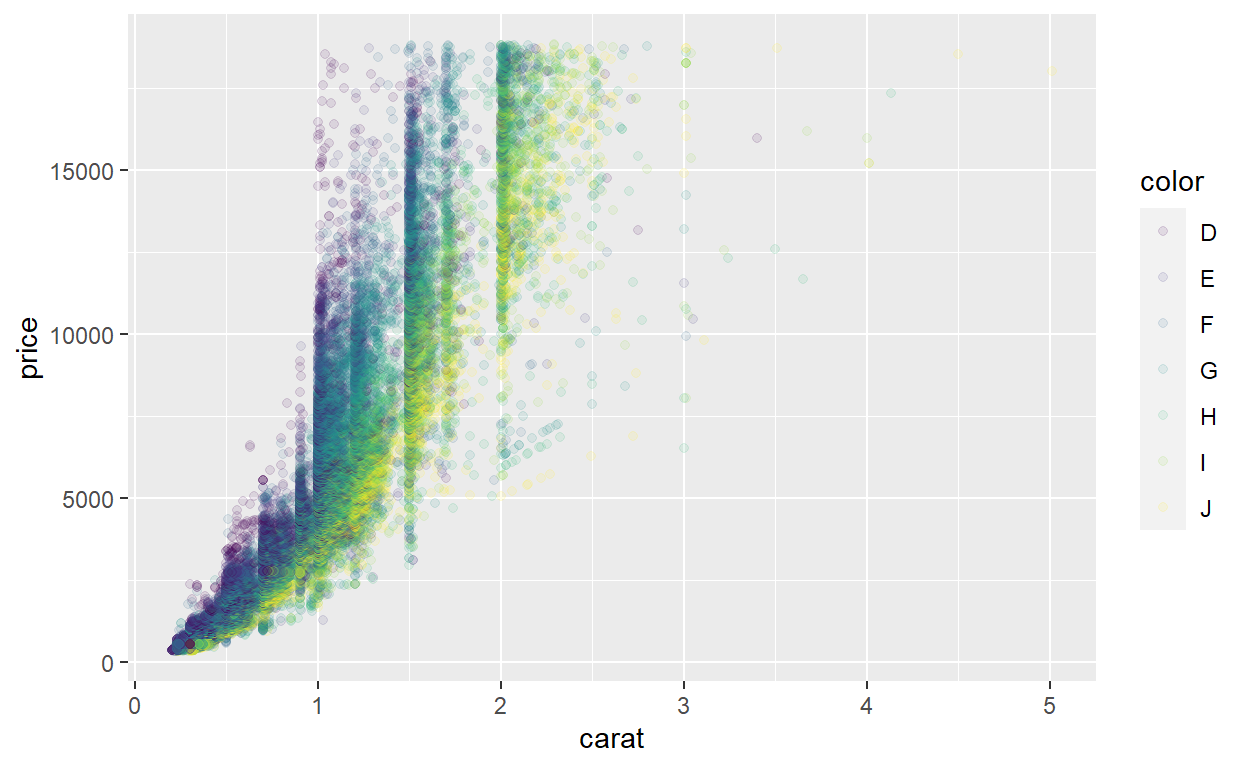
## Aesthetics/attributes para un *scatterplot*

geom\_point entiende los siguientes *aesthetics*:

* x, y.
* colour: Color **del borde**.
* shape: Forma del punto.
* fill: Color del relleno.
* alpha: mide la transparencia (inverso del número de objetos que han de superponerse para lograr opacidad).
* size: Tamaño del punto.

## *Scatterplots* algo más vistosos (I)

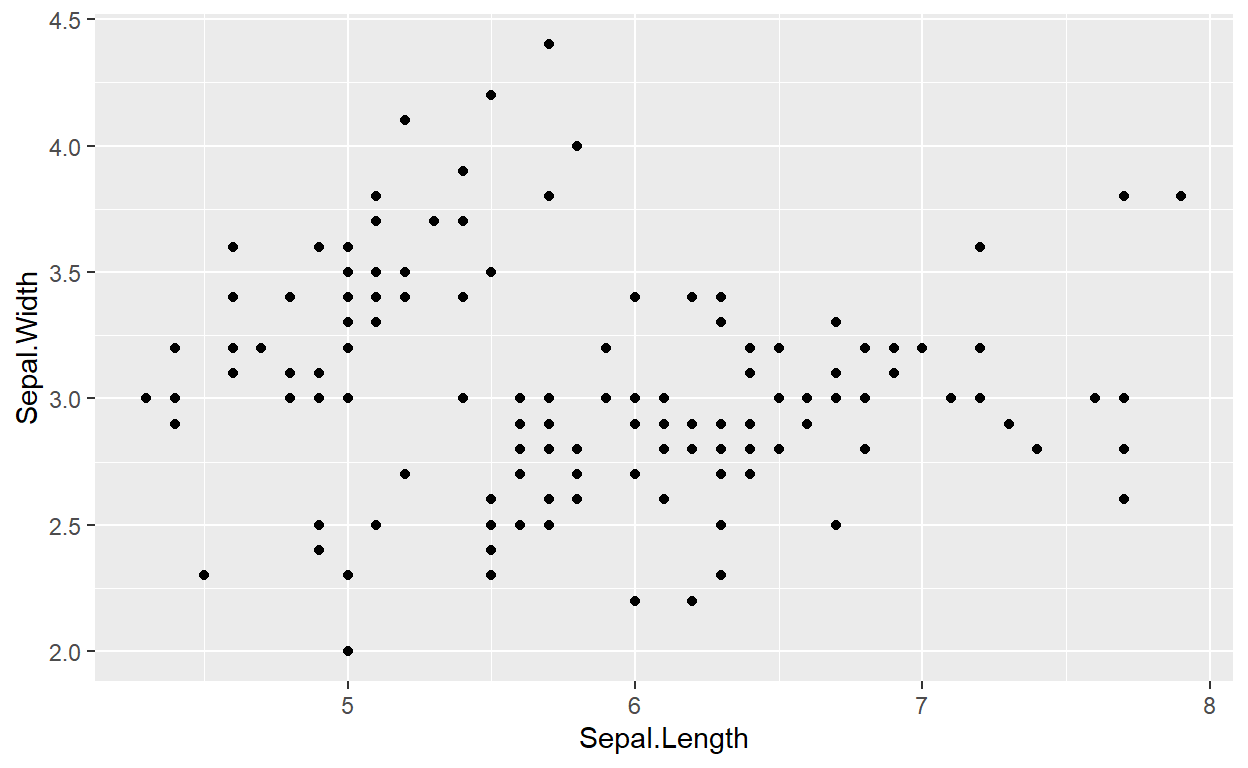
ggplot(data = diamonds, aes(x = carat, y = price, col = color)) +   
 geom\_point(alpha = 1/10)



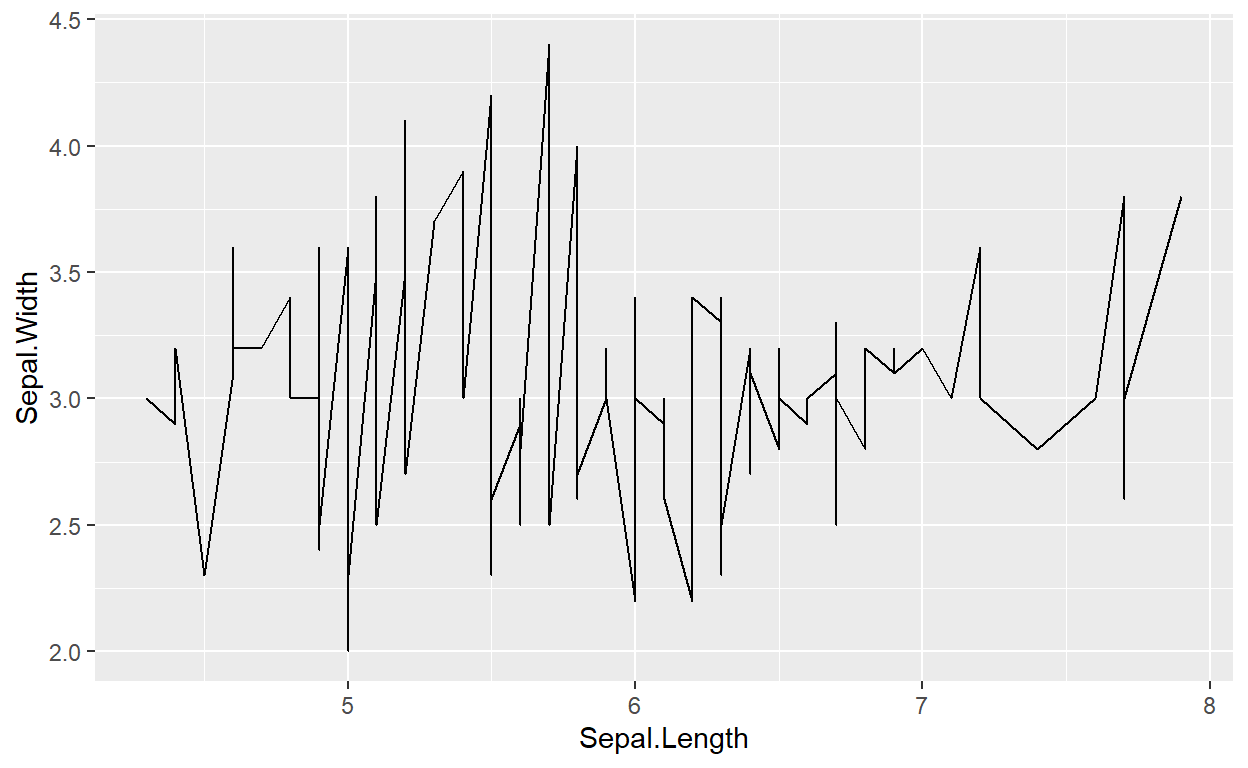
## 3. Geoms

Las capas (o **geoms** para **ggplot2**) son los verbos del lenguaje de los gráficos. Indican qué hacer con los datos y las estéticas elegidas.

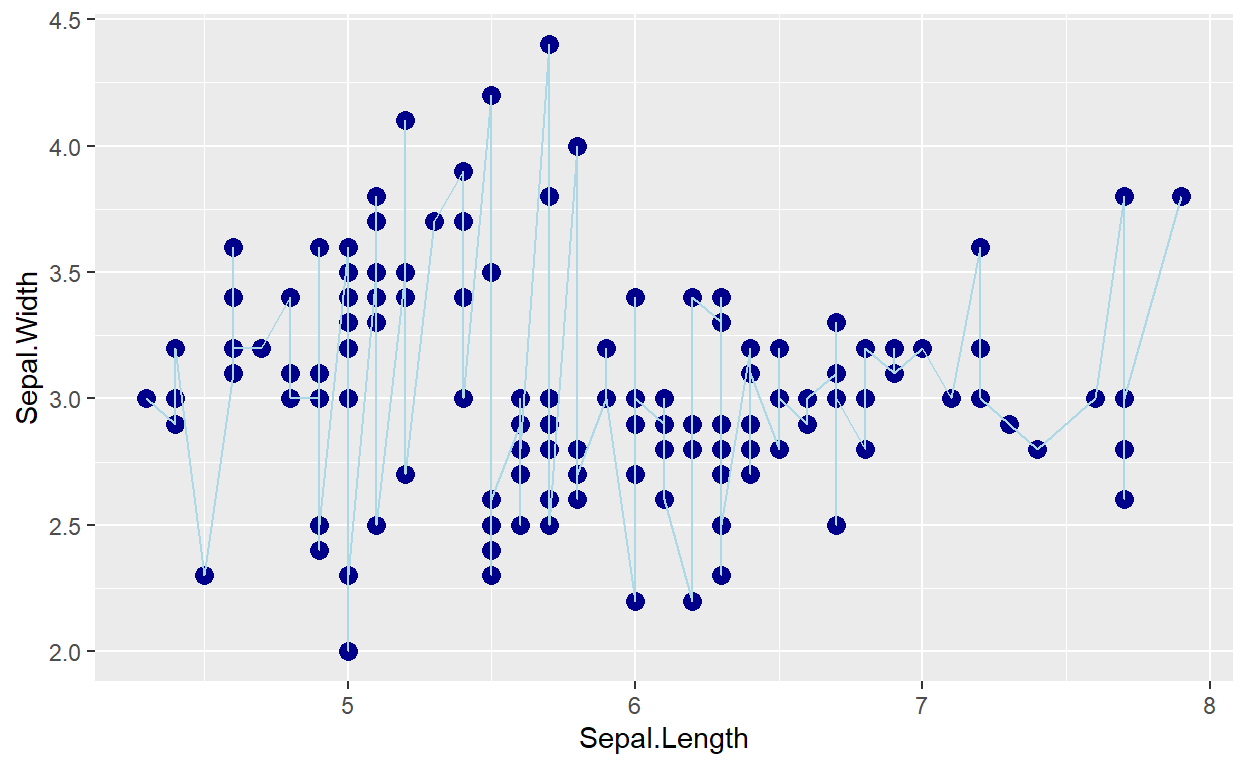
data(iris)   
  
gg <- ggplot(data = iris, aes(x = Sepal.Length, y = Sepal.Width))   
  
gg + geom\_point()



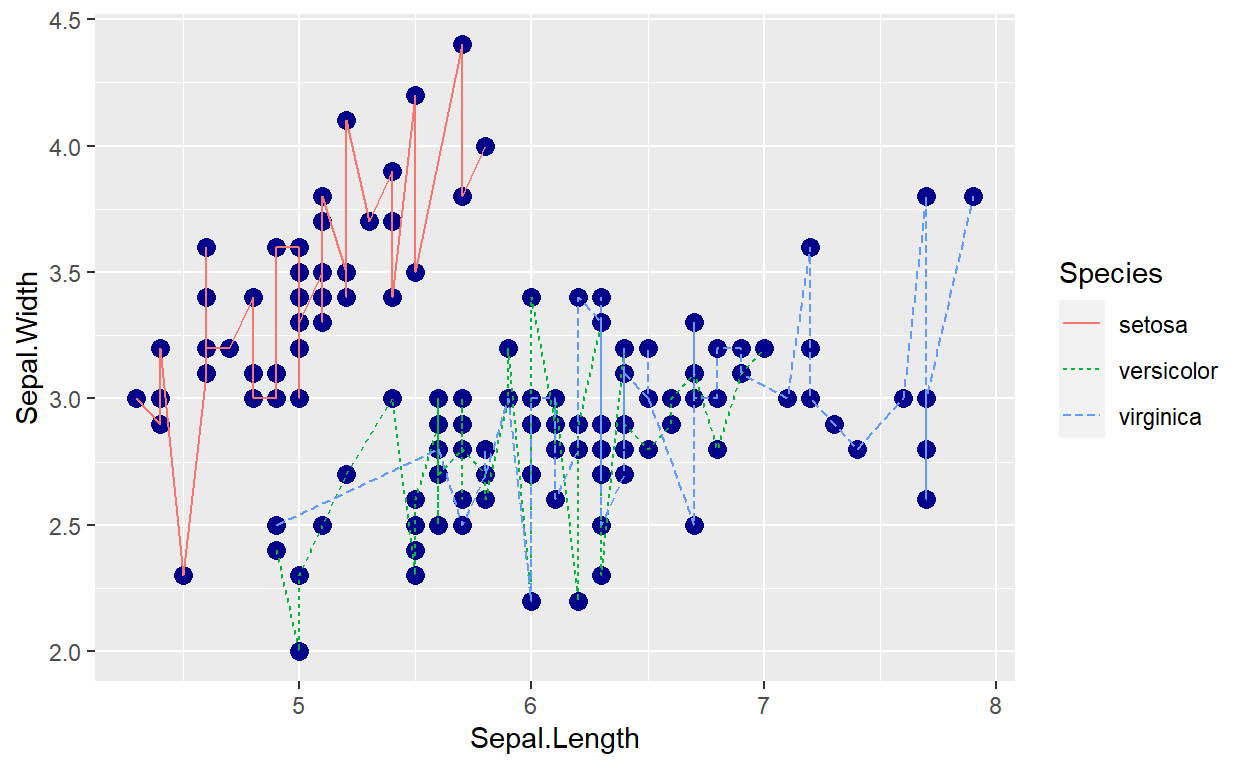
gg + geom\_line()



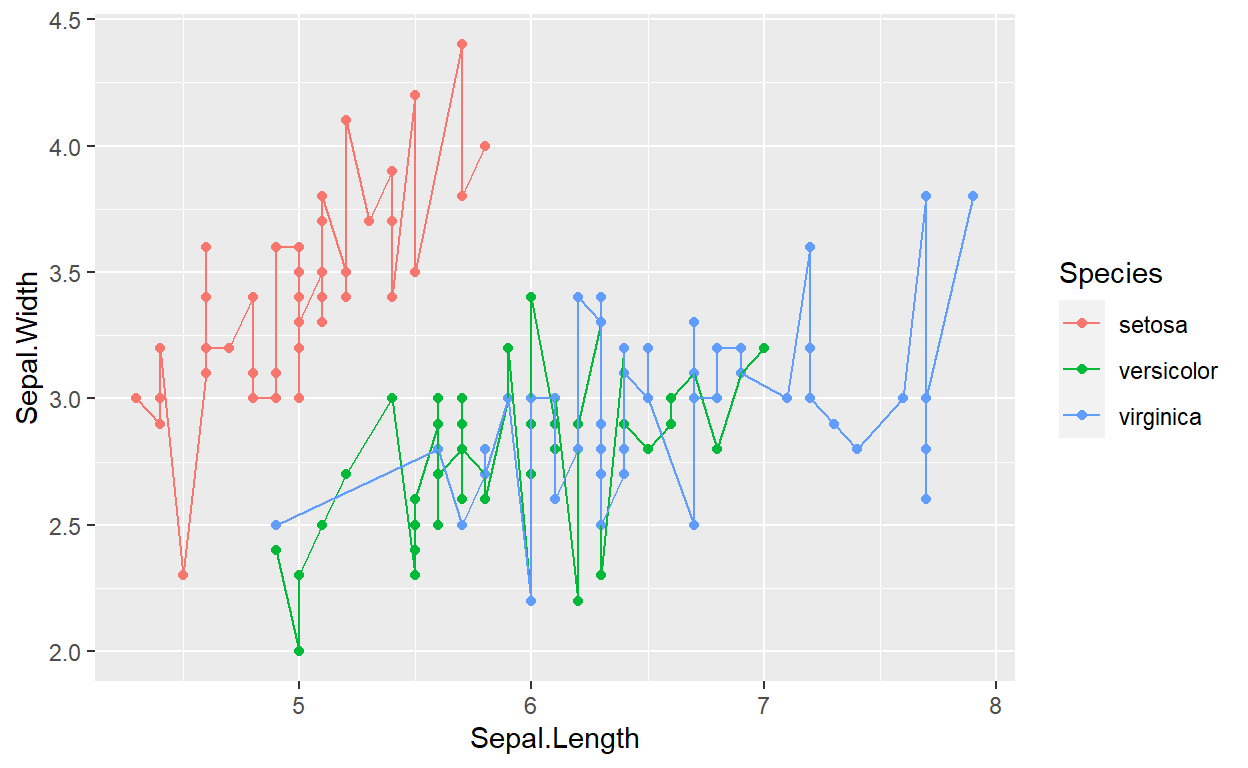
gg + geom\_point(colour = 'darkblue', size = 3) + geom\_line(colour = 'lightblue')



gg + geom\_point(colour = 'darkblue', size = 3) +  
 geom\_line(aes(colour = Species, linetype = Species))



ggplot(data = iris, aes(x = Sepal.Length, y = Sepal.Width,colour = Species)) +  
 geom\_point() + geom\_line()



## Un nuevo set de datos

Antes de seguir con los componentes de un gráfico, vamos a echar un vistazo a un nuevo set de datos y los gráficos estadísticos más habituales.

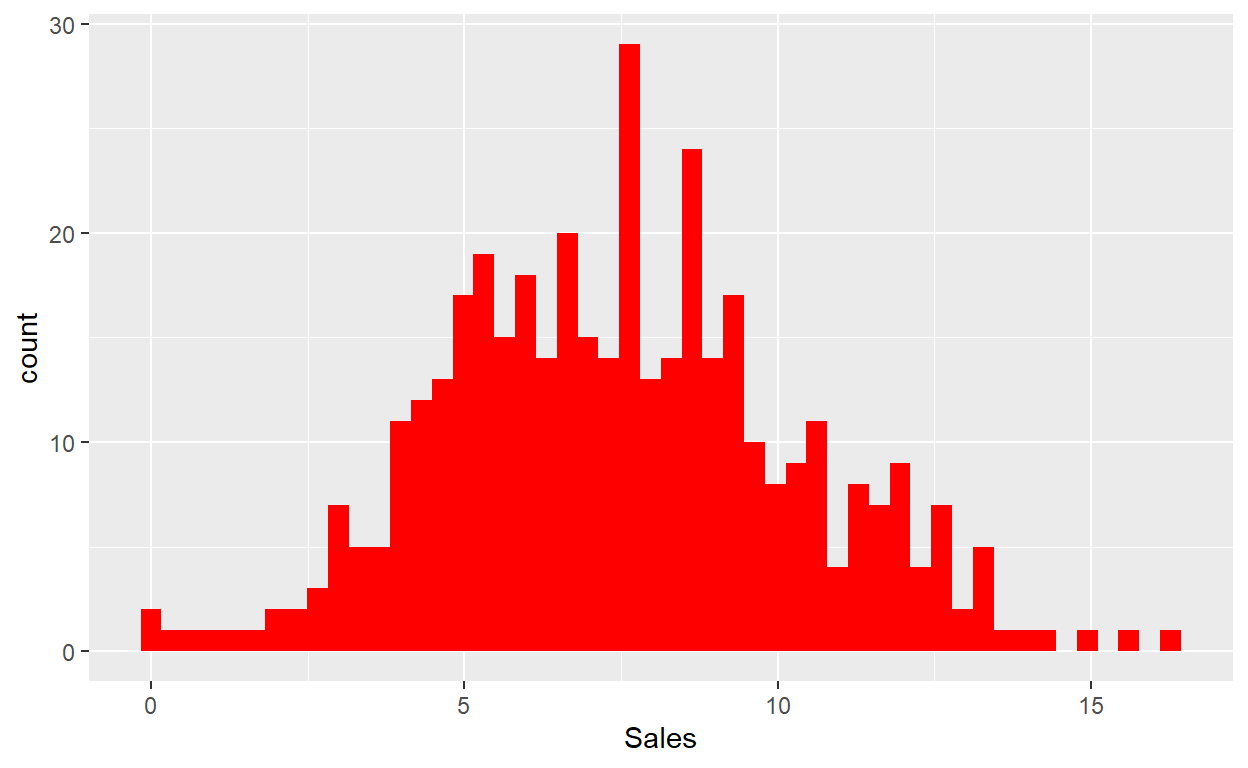
#install.packages('ISLR', dependencies=TRUE, repos='http://cran.rstudio.com/')  
library(ISLR)  
data(Carseats)  
str(Carseats)

## 'data.frame': 400 obs. of 11 variables:  
## $ Sales : num 9.5 11.22 10.06 7.4 4.15 ...  
## $ CompPrice : num 138 111 113 117 141 124 115 136 132 132 ...  
## $ Income : num 73 48 35 100 64 113 105 81 110 113 ...  
## $ Advertising: num 11 16 10 4 3 13 0 15 0 0 ...  
## $ Population : num 276 260 269 466 340 501 45 425 108 131 ...  
## $ Price : num 120 83 80 97 128 72 108 120 124 124 ...  
## $ ShelveLoc : Factor w/ 3 levels "Bad","Good","Medium": 1 2 3 3 1 1 3 2 3 3 ...  
## $ Age : num 42 65 59 55 38 78 71 67 76 76 ...  
## $ Education : num 17 10 12 14 13 16 15 10 10 17 ...  
## $ Urban : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 2 2 1 1 ...  
## $ US : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 2 1 2 1 2 ...

Documentación: [aquí](https://cran.r-project.org/web/packages/ISLR/ISLR.pdf).

## Gráficos univariantes: histograma (I)

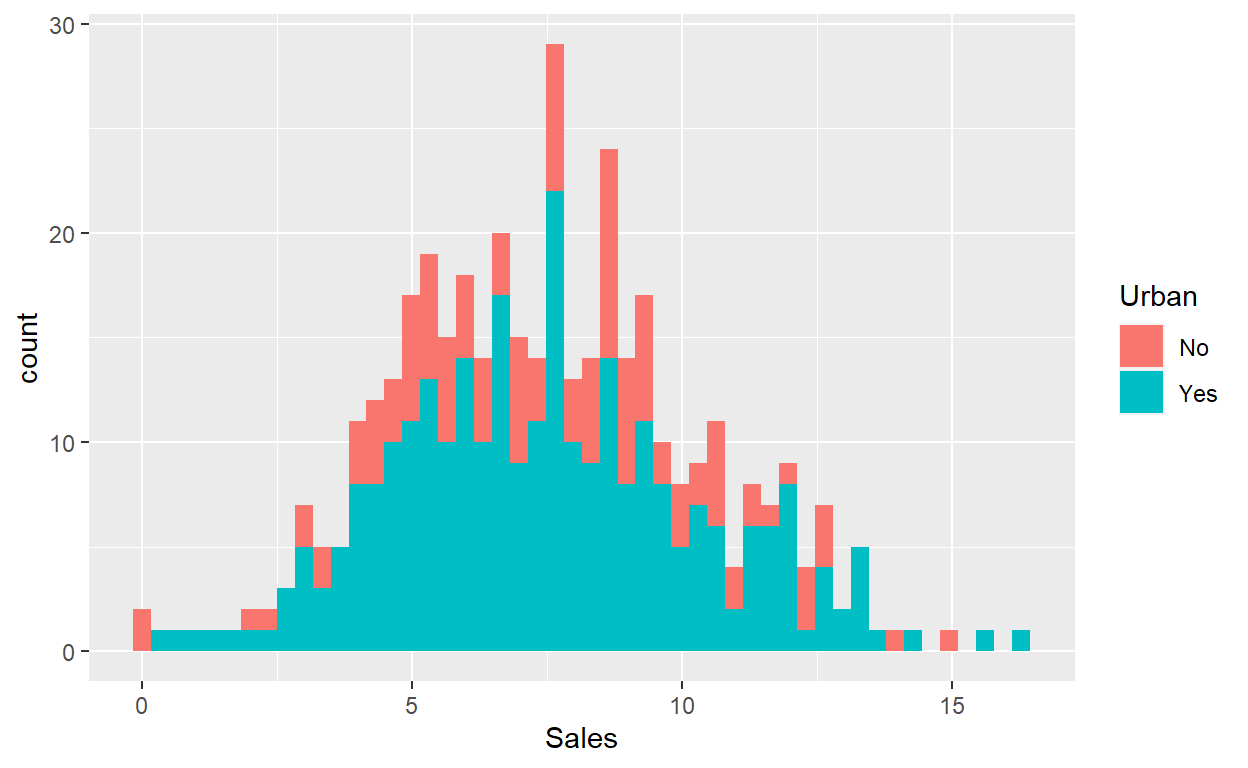
ggplot(data = Carseats, aes(x = Sales)) +   
 geom\_histogram(bins = 50, fill = 'red')



# El parámetro 'bins' puede sustituirse por 'binwidth'

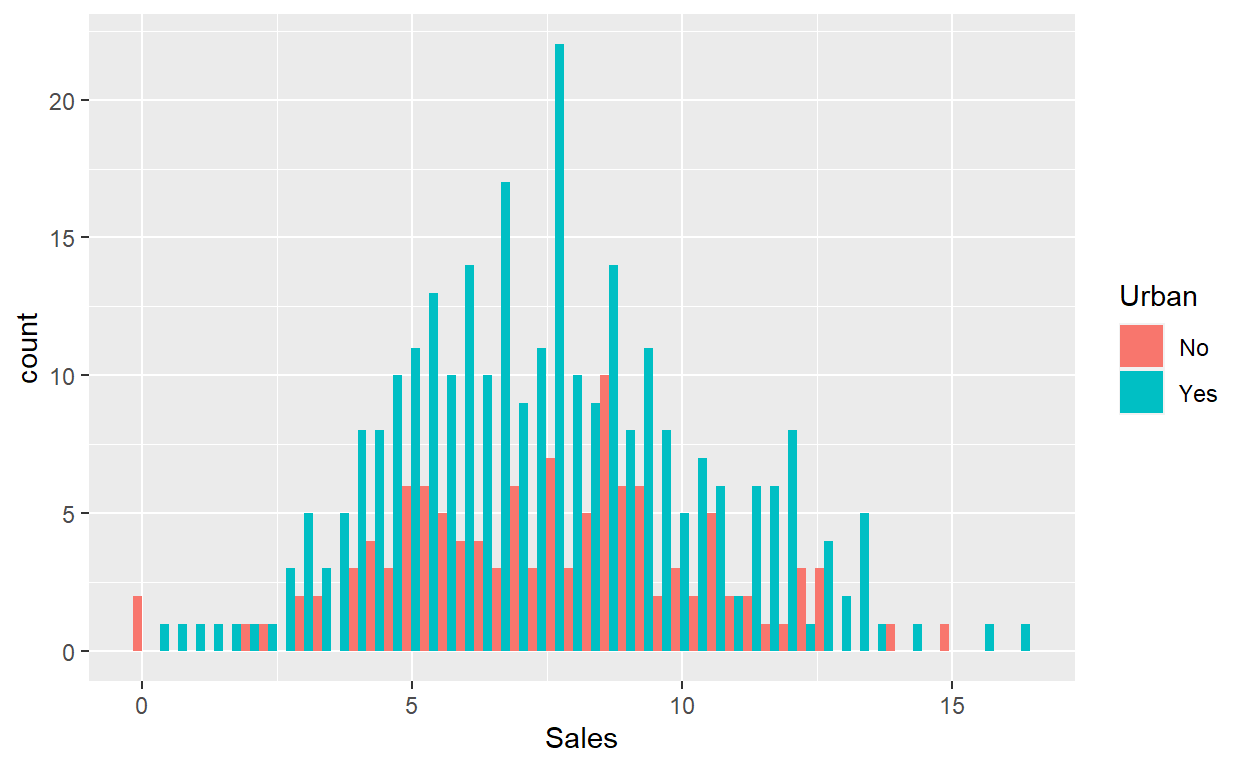
## Gráficos univariantes: histograma (II)

ggplot(data = Carseats, aes(x = Sales, fill = Urban)) +   
 geom\_histogram(bins = 50)



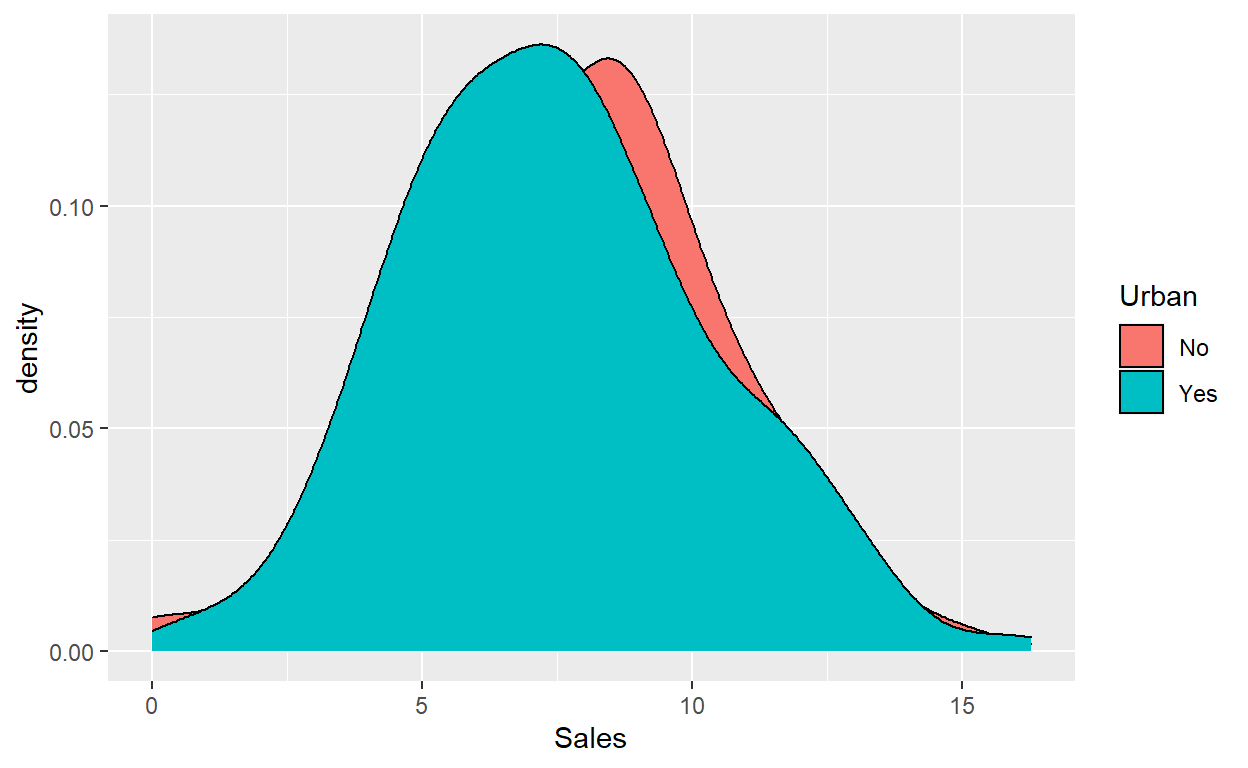
## Gráficos univariantes: histograma (III)

ggplot(data = Carseats, aes(x = Sales, fill = Urban)) +   
 geom\_histogram(position = 'dodge', bins = 50)



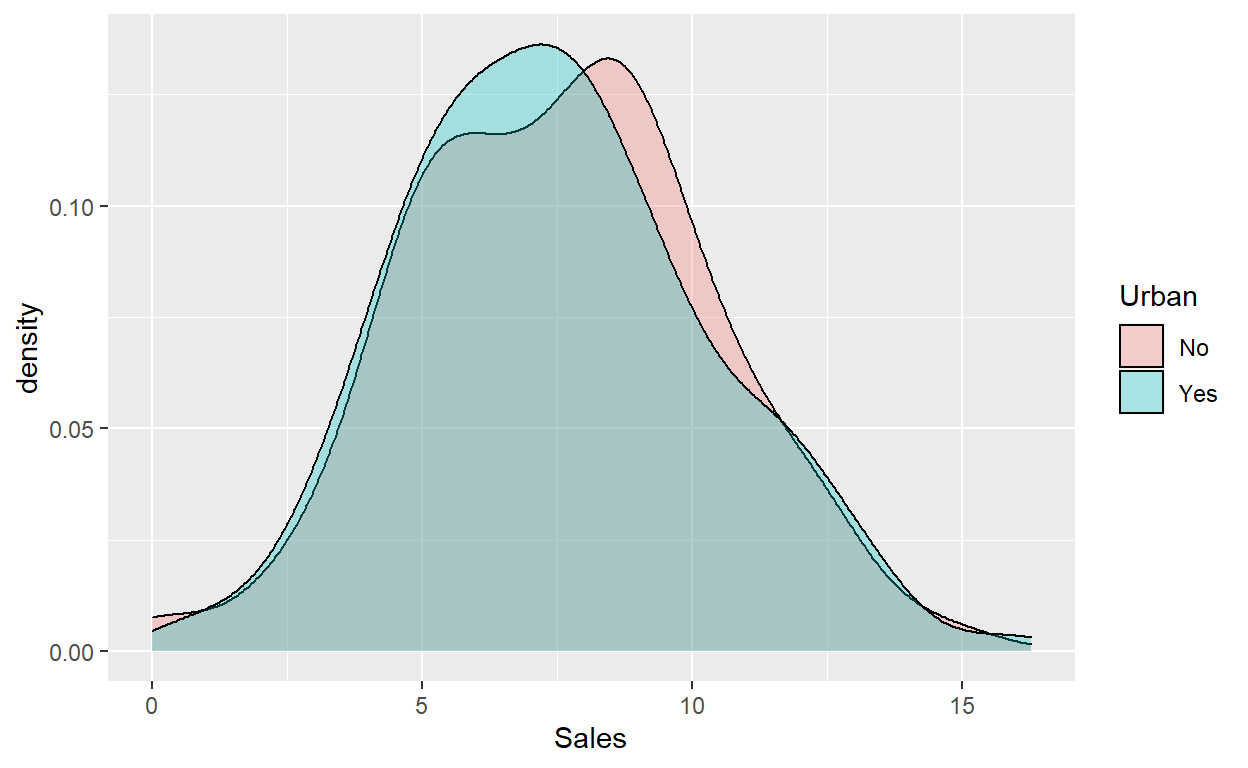
## Gráficos univariantes: densidad (I)

ggplot(data = Carseats, aes(x = Sales, fill = Urban)) +   
 geom\_density()



## Gráficos univariantes: densidad (II)

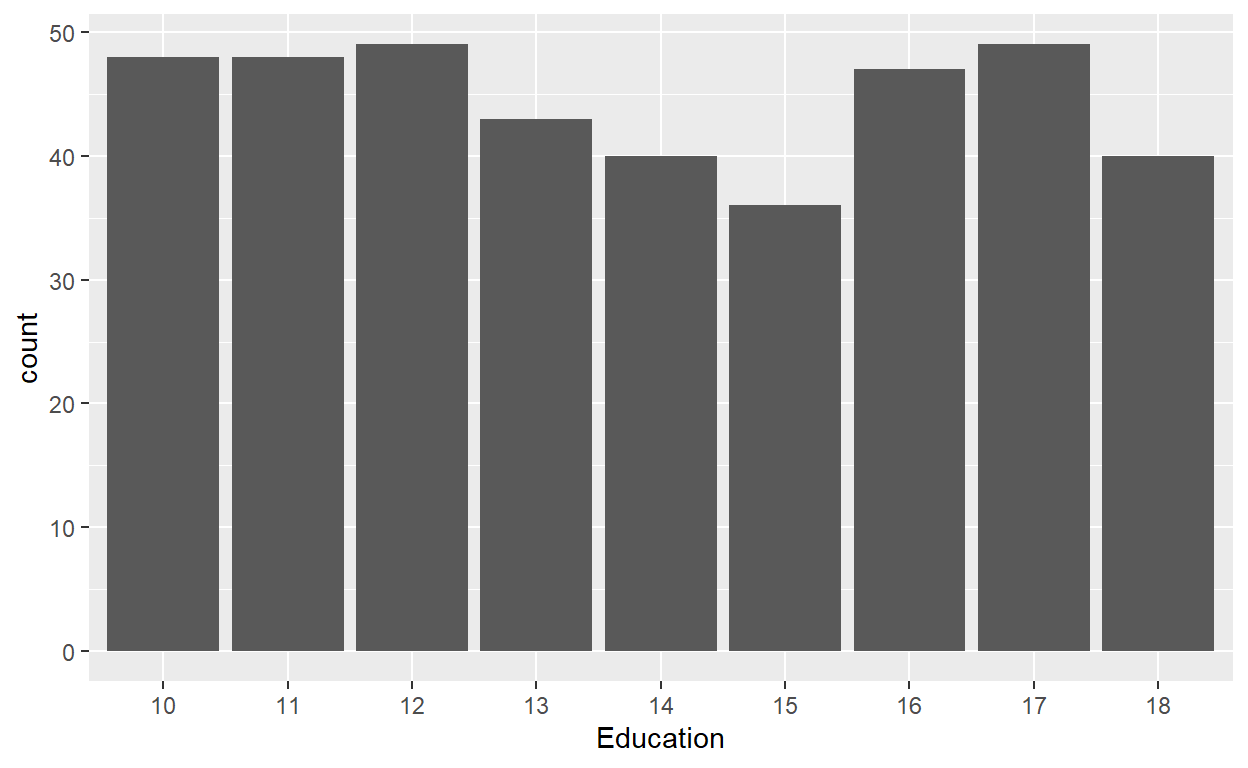
ggplot(data = Carseats, aes(x = Sales, fill = Urban)) +   
 geom\_density(alpha = 0.3)



## Gráficos univariantes: barras (I)

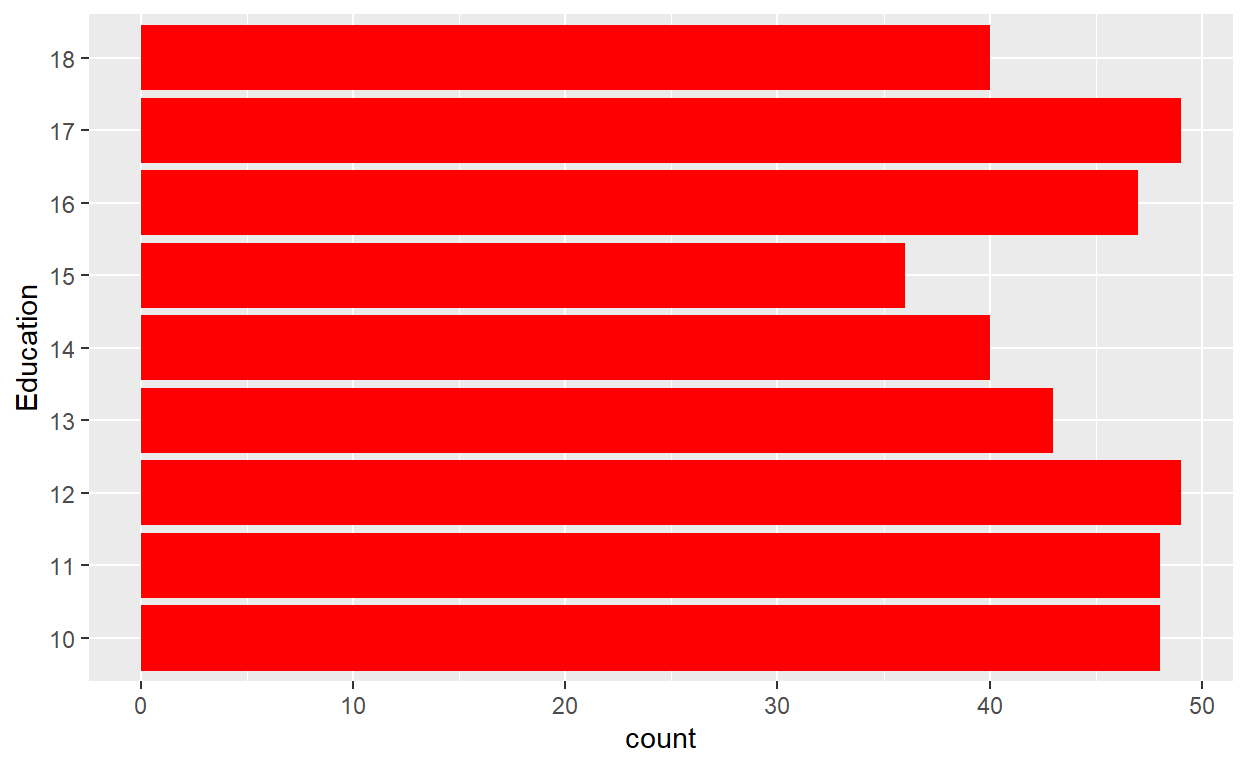
Carseats$Education <- as.ordered(Carseats$Education)

ggplot(data = Carseats, aes(x = Education)) + geom\_bar()



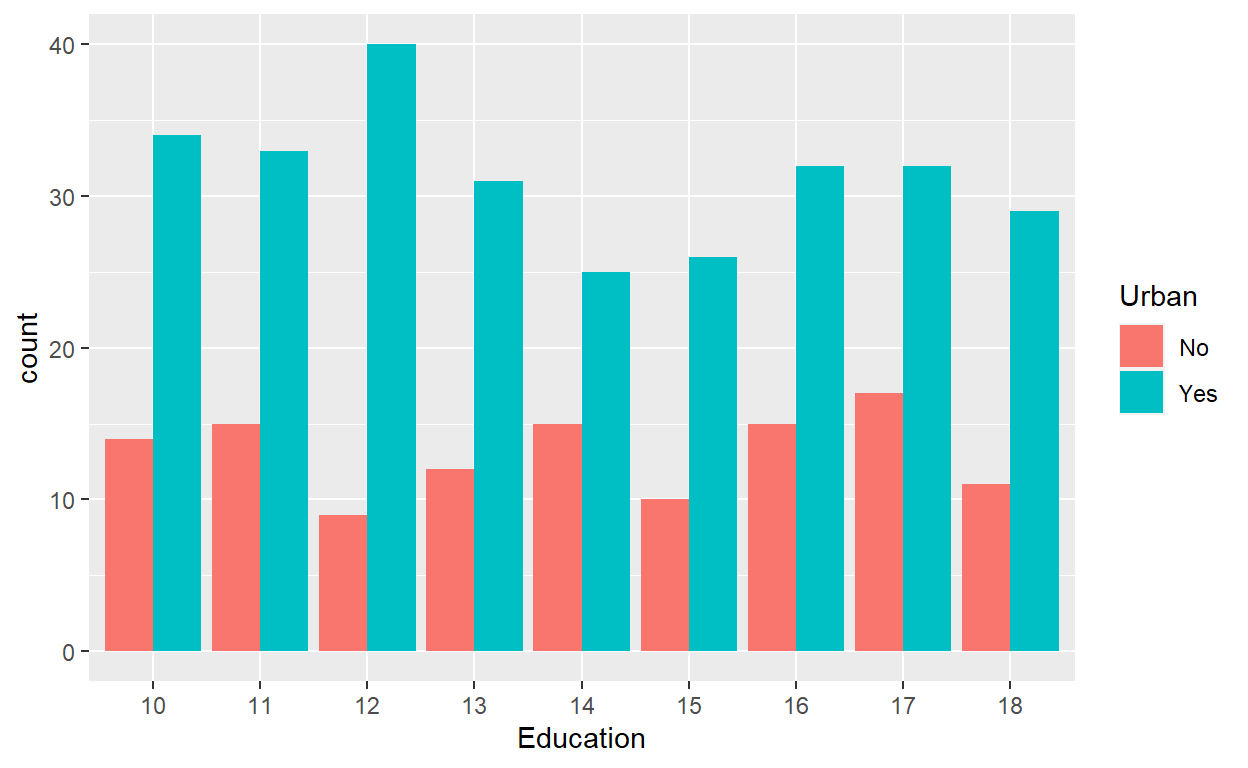
## Gráficos univariantes: barras (II)

ggplot(data = Carseats, aes(x = Education)) +   
 geom\_bar(fill = 'red') +   
 coord\_flip()



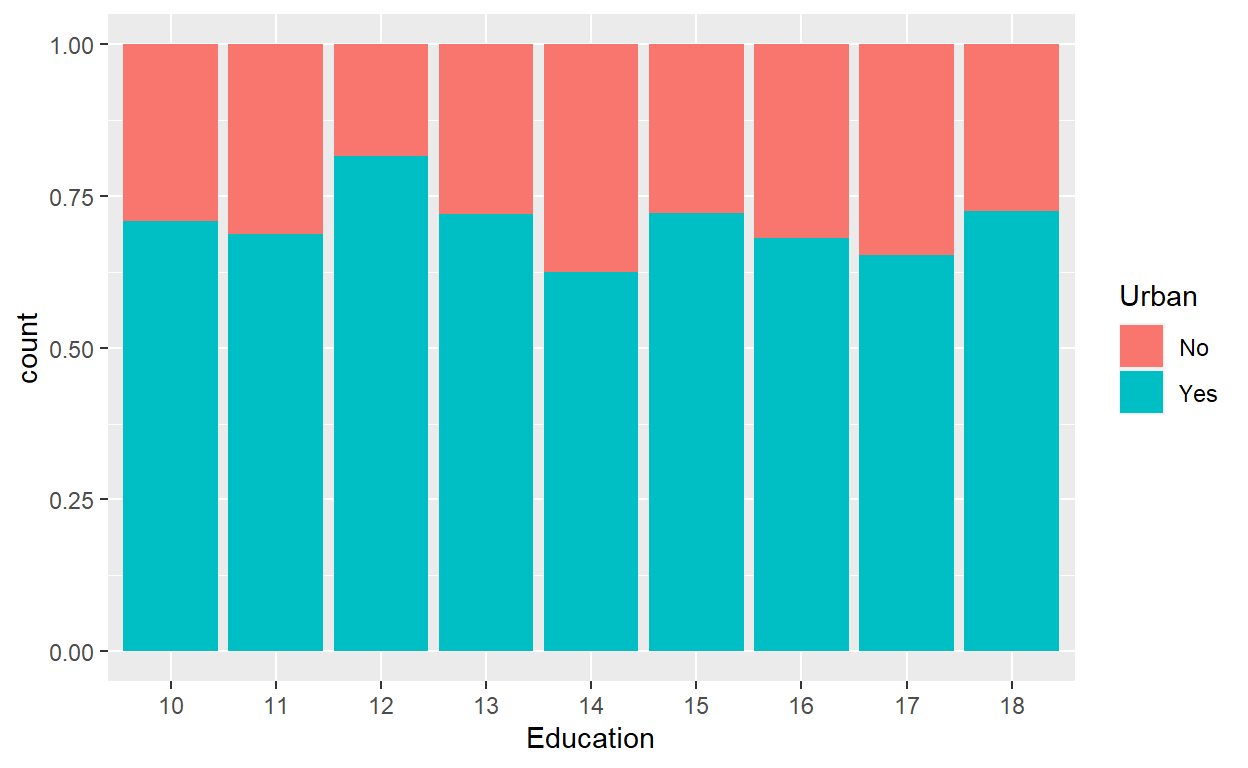
## Gráficos univariantes: barras (III)

ggplot(data = Carseats, aes(x = Education, fill = Urban)) +   
 geom\_bar(position = 'dodge')



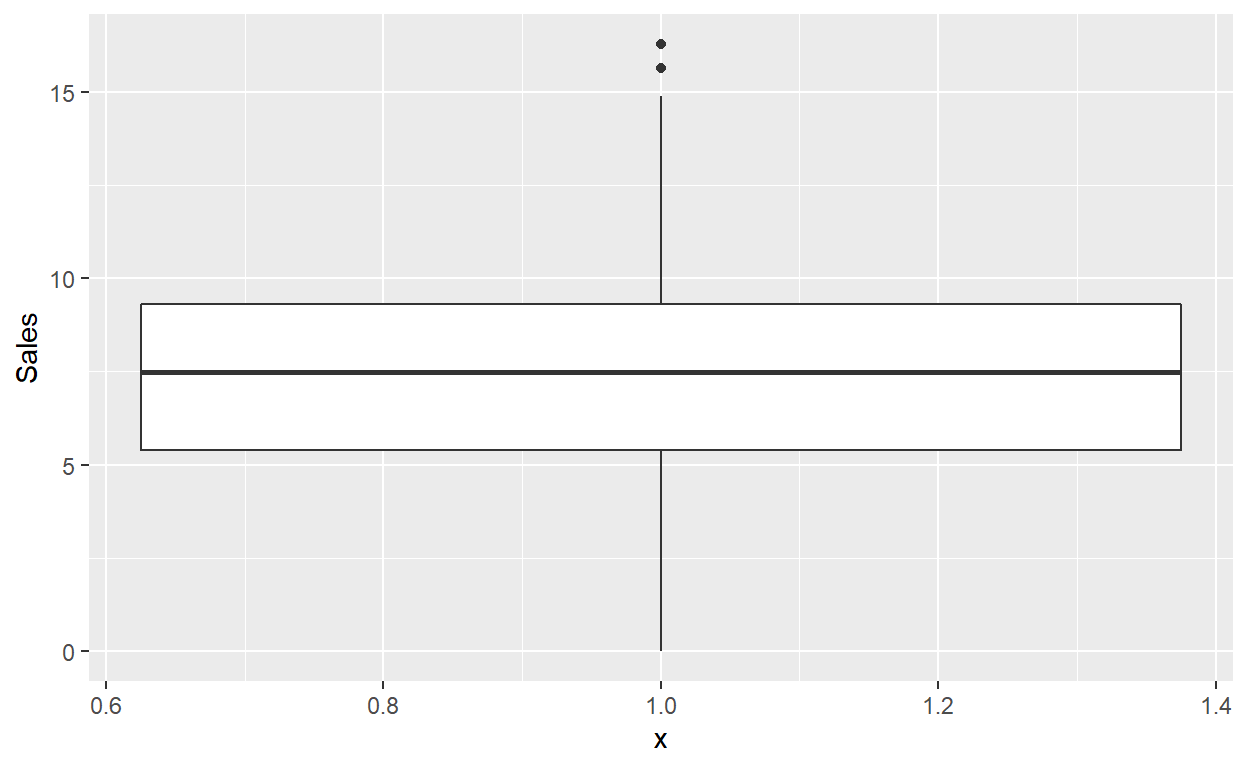
## Gráficos univariantes: barras (IV)

ggplot(data = Carseats, aes(x = Education, fill = Urban)) +   
 geom\_bar(position = 'fill')



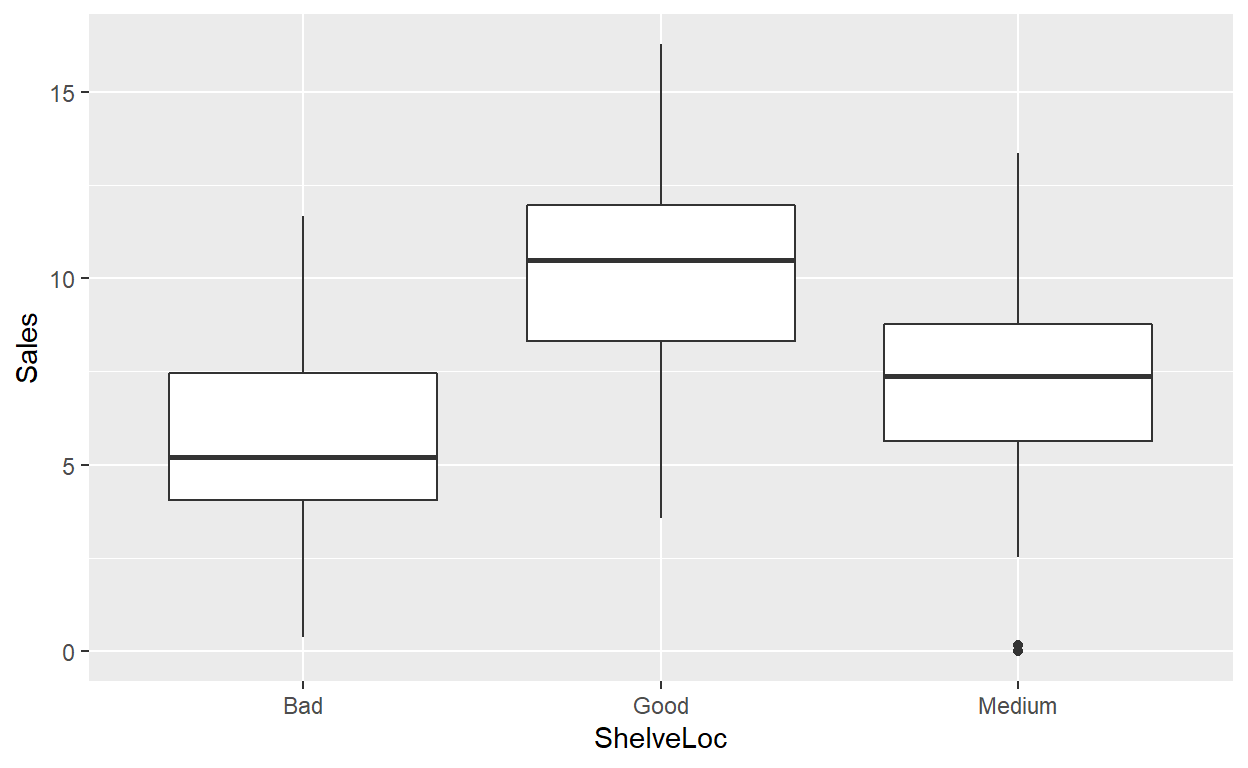
## Gráficos univariantes: boxplots (I)

ggplot(data = Carseats, aes(x = 1, y = Sales)) +   
 geom\_boxplot()



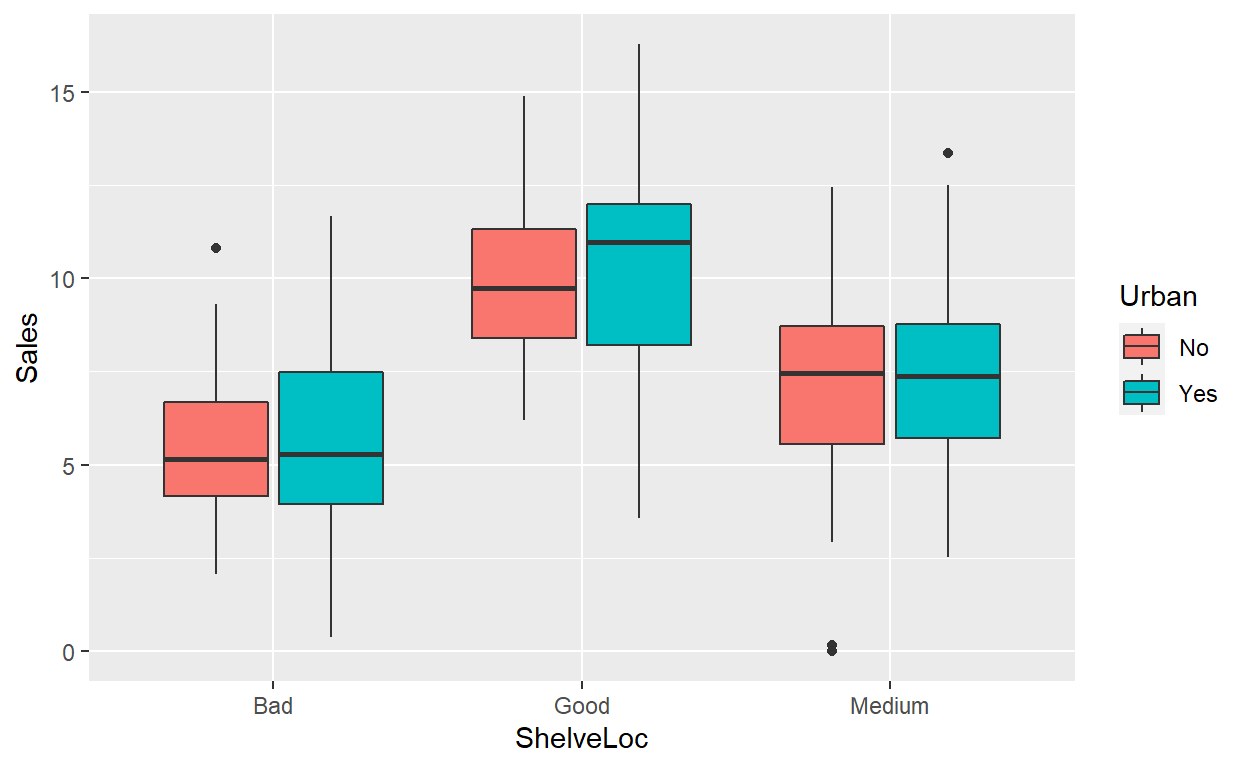
## Gráficos univariantes: boxplots (II)

ggplot(data = Carseats, aes(x = ShelveLoc, y = Sales)) +   
 geom\_boxplot()



## Gráficos univariantes: boxplots (III)

ggplot(data = Carseats, aes(x = ShelveLoc, y = Sales, fill = Urban)) +   
 geom\_boxplot()



## 4. Stats (I)

Las transformaciones estadísticas (*stat*) resumen los datos en distintas formas. Por ejemplo, conteos e intervalos en los histogramas.

Sea explícita o implícitamente, todos los gráficos de ggplot2 contienen un elemento obligatorio más: los *stats*. Los *stats* son agregaciones de los datos que facilitan su presentación y se asocian a cada *geom*.

* Los puntos, que no representan ningún tipo de agregación, llevan por defecto el *stat* ***‘identity’***.
* La línea de tendencia, que se define mediante la agregación dada por el modelo GAM, viene dada por el *stat* ***‘smooth’***.

## 4. Stats (II)

¿Qué muestra este gráfico de barras?

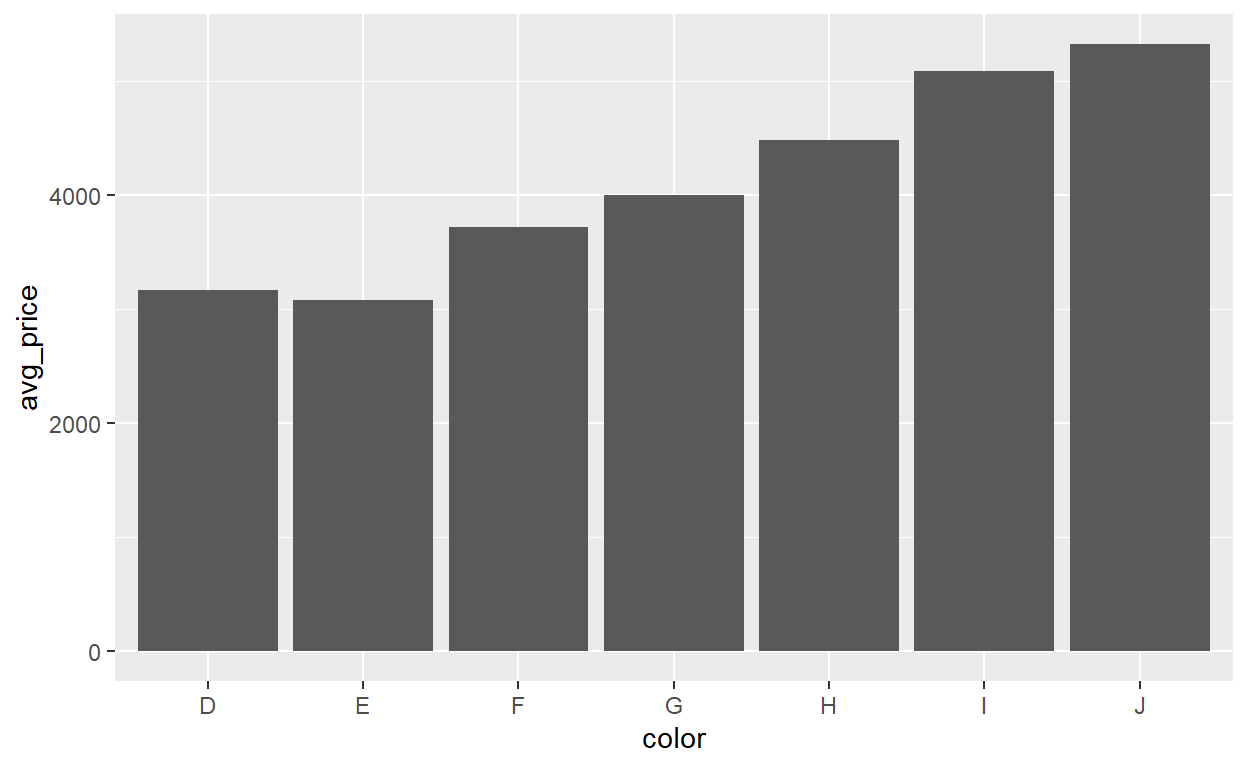
ggplot(data = diamonds, aes(x = color)) + geom\_bar()



## 4. Stats (III)

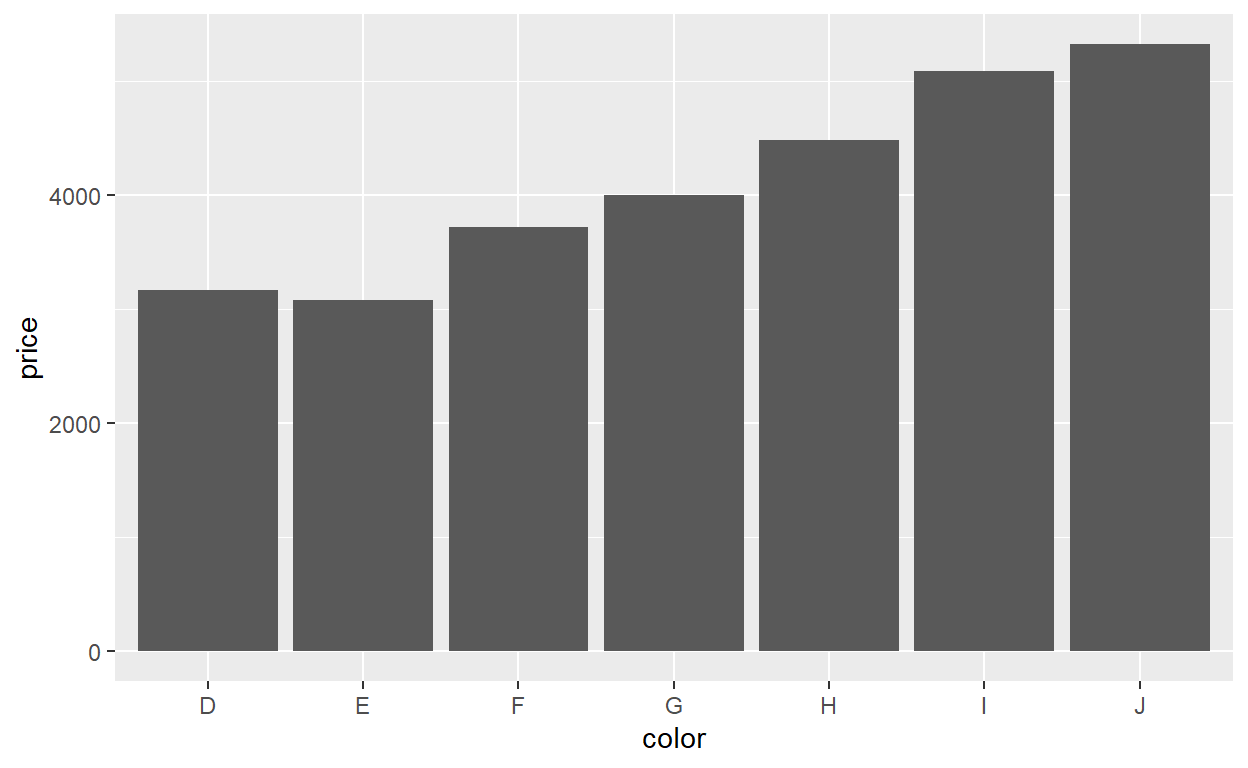
¿Qué hacemos si queremos mostrar el precio medio por color en un gráfico de barras?

diamonds %>%   
 group\_by(color) %>%   
 summarise(avg\_price = mean(price)) %>%   
 ggplot(aes(x = color, y = avg\_price)) +  
 geom\_bar(stat = 'identity')



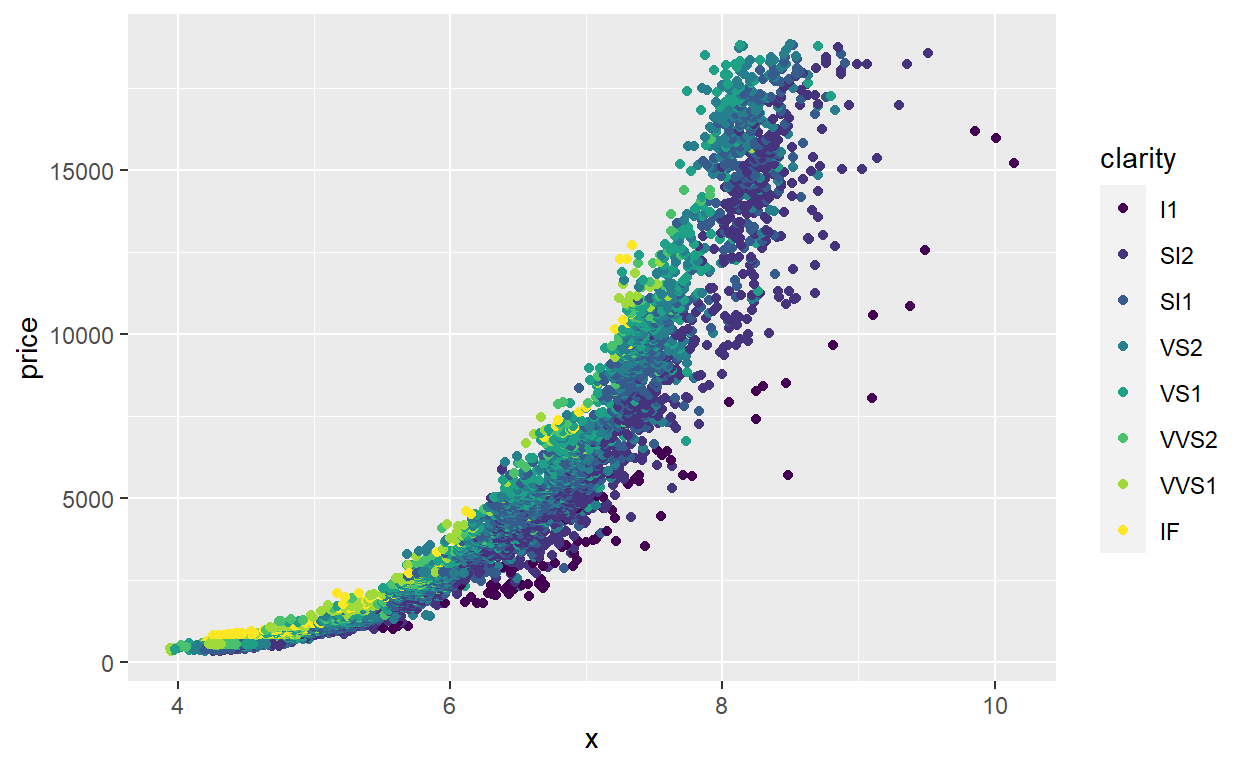
## 4. Stats (IV)

ggplot(data = diamonds, aes(x = color, y = price)) +  
 stat\_summary(fun.y = 'mean', geom = 'bar')



## 4. Stats (V)

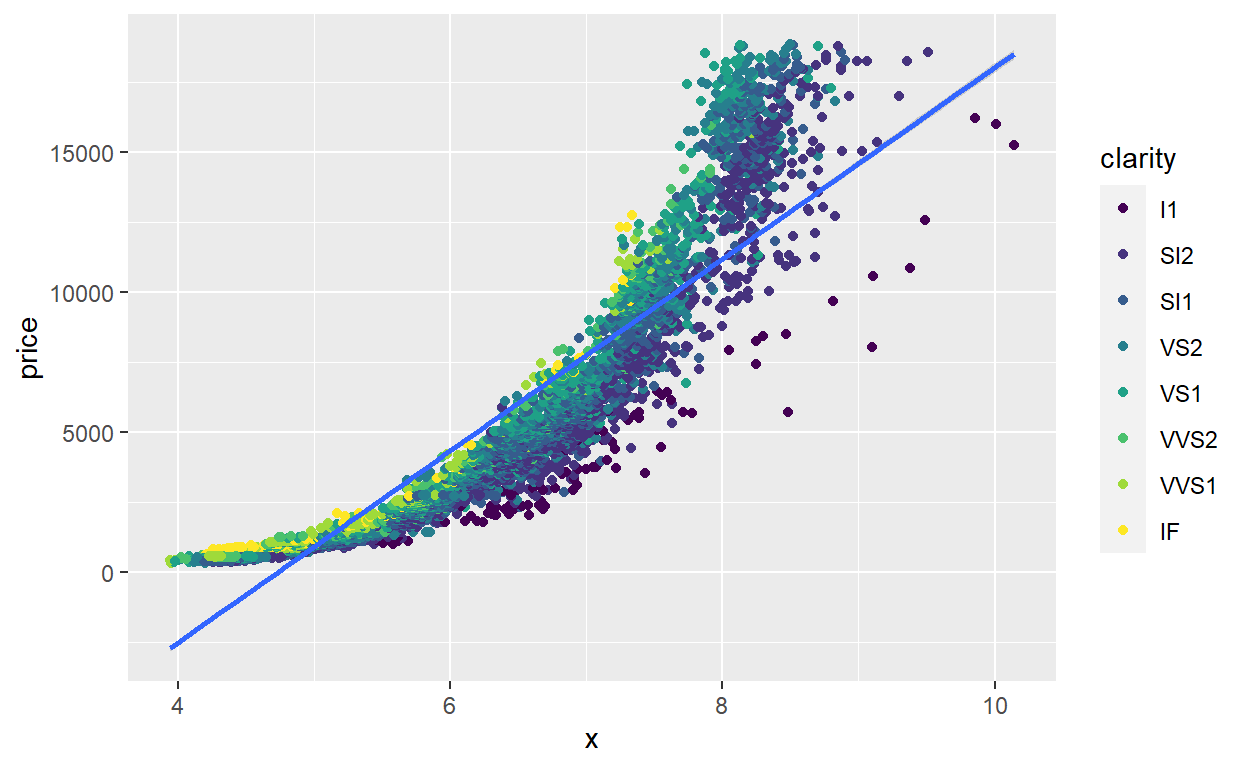
ggplot(data = diamonds[diamonds$color == 'I',], aes(x = x, y = price)) +  
 geom\_point(aes(colour = clarity))



## 4. Stats (VI)

ggplot(data = diamonds[diamonds$color == 'I',], aes(x = x, y = price)) +  
 geom\_point(aes(colour = clarity)) +  
 stat\_smooth(method = 'lm')

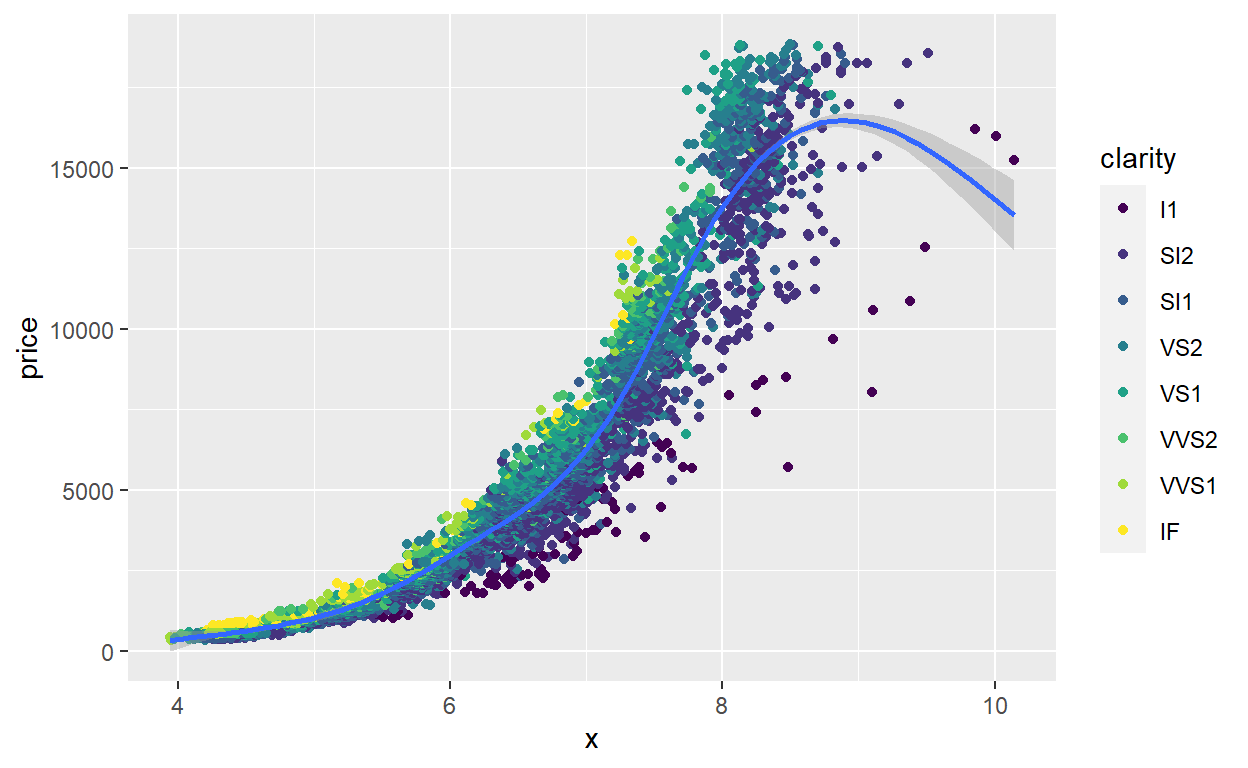
## `geom\_smooth()` using formula 'y ~ x'



## 4. Stats (VII)

ggplot(data = diamonds[diamonds$color == 'I',], aes(x = x, y = price)) +  
 geom\_point(aes(colour = clarity)) +  
 stat\_smooth()

## `geom\_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'



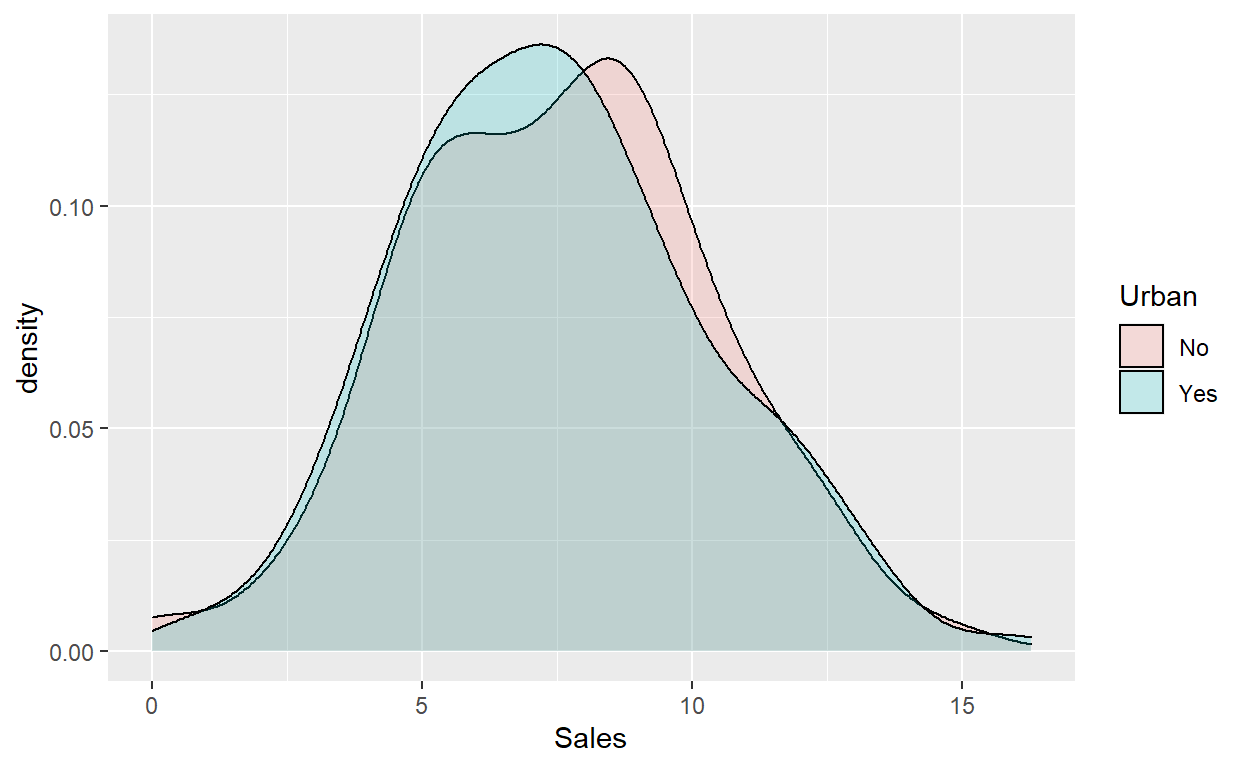
## 5. Facets (I)

Los *facets* permiten separar gráficos “superpuestos” en base a valores de una variable categórica. El *facetting* es mucho más eficiente que hacer varios gráficos por separado y unirlos.

library(ISLR)  
data(Carseats)  
str(Carseats)

## 'data.frame': 400 obs. of 11 variables:  
## $ Sales : num 9.5 11.22 10.06 7.4 4.15 ...  
## $ CompPrice : num 138 111 113 117 141 124 115 136 132 132 ...  
## $ Income : num 73 48 35 100 64 113 105 81 110 113 ...  
## $ Advertising: num 11 16 10 4 3 13 0 15 0 0 ...  
## $ Population : num 276 260 269 466 340 501 45 425 108 131 ...  
## $ Price : num 120 83 80 97 128 72 108 120 124 124 ...  
## $ ShelveLoc : Factor w/ 3 levels "Bad","Good","Medium": 1 2 3 3 1 1 3 2 3 3 ...  
## $ Age : num 42 65 59 55 38 78 71 67 76 76 ...  
## $ Education : num 17 10 12 14 13 16 15 10 10 17 ...  
## $ Urban : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 2 2 1 1 ...  
## $ US : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 2 1 2 1 2 ...

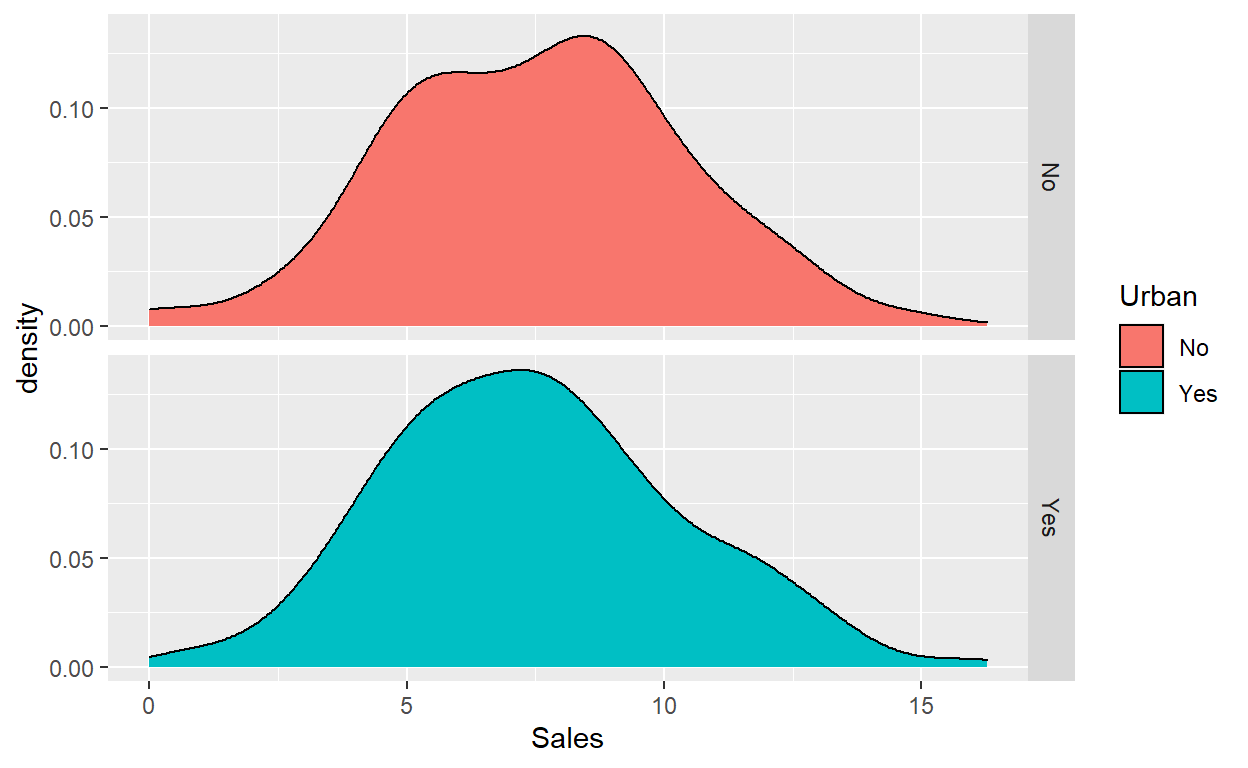
ggplot(data = Carseats, aes(x = Sales, fill = Urban)) +   
 geom\_density(alpha = 0.2)



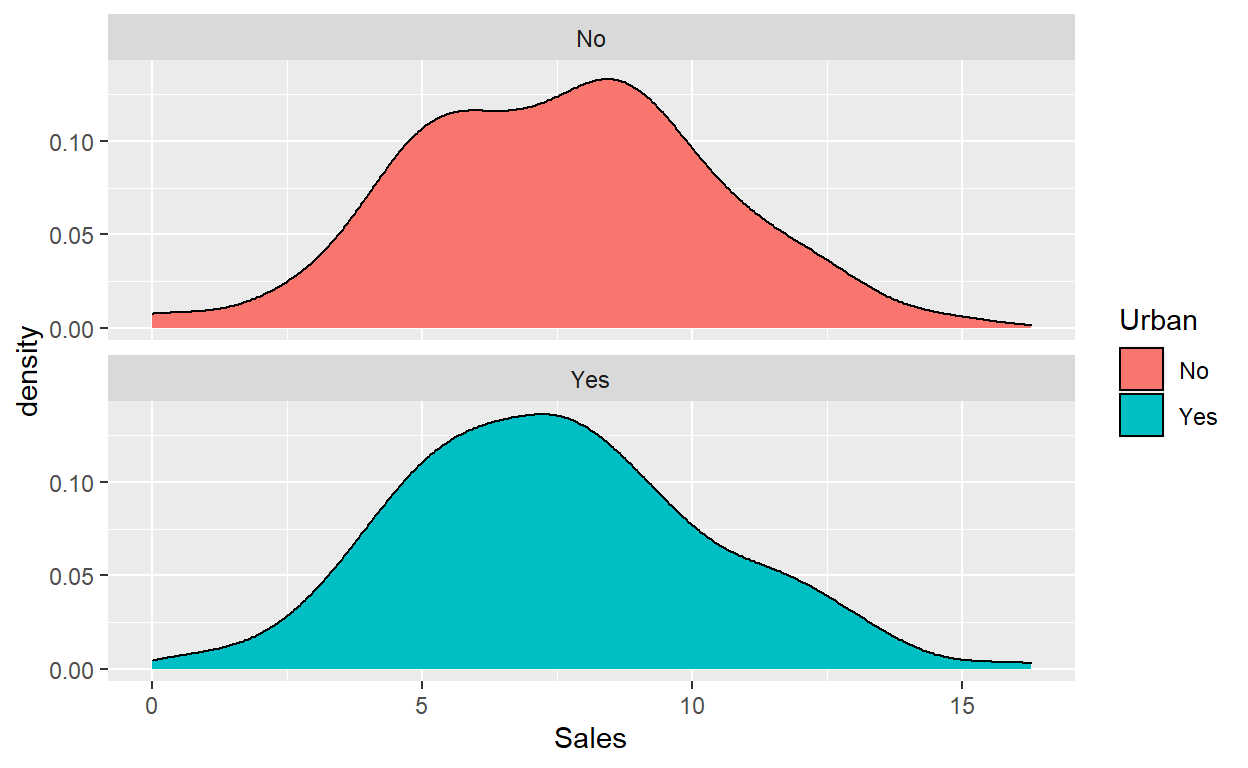
## 5. Facets (II)

**Dividir por factores.**

ggplot(data = Carseats, aes(x = Sales, fill = Urban)) +   
 geom\_density() +   
 facet\_grid(Urban ~ .)



ggplot(data = Carseats, aes(x = Sales, fill = Urban)) +   
 geom\_density() +   
 facet\_wrap(~Urban, nrow = 2)

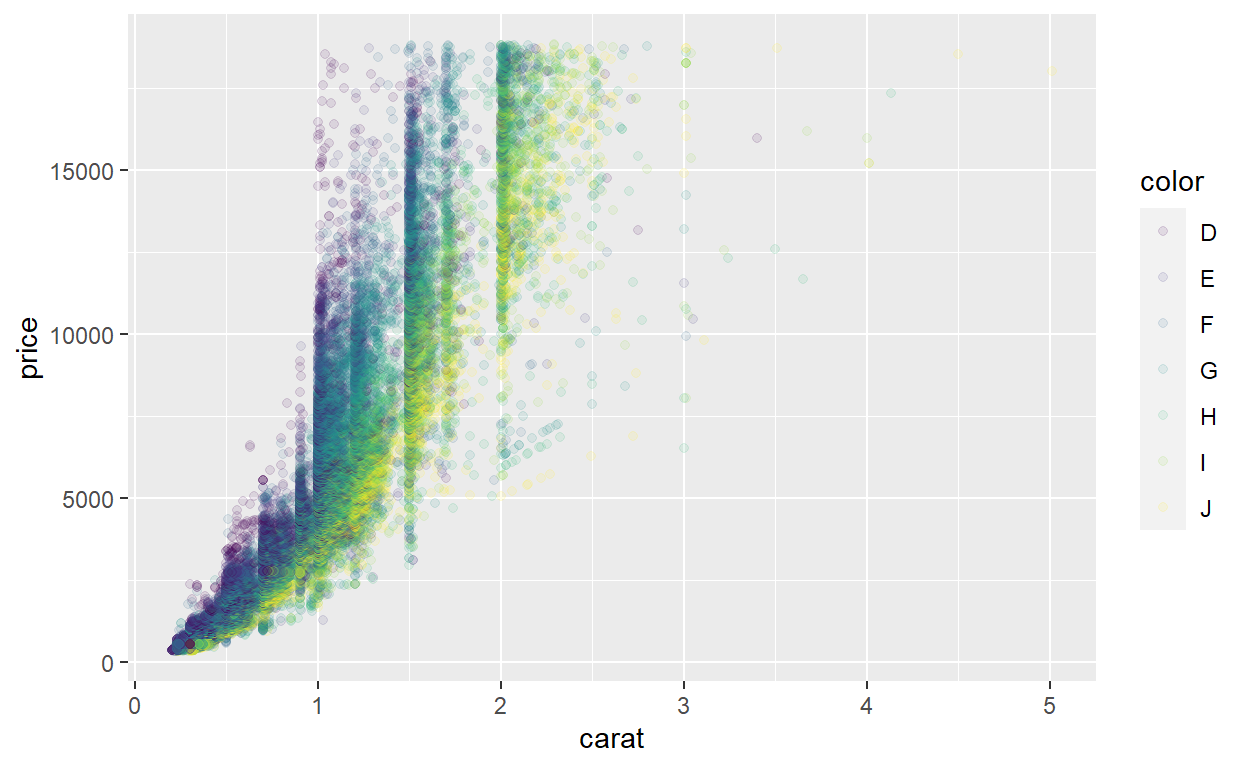


## 6.Escalas (I)

Las escalas (*scale*) mapean valores específicos de una variable del set de datos a los atributos estéticos (*aesthetic*: color, tamaño o forma). Concretamente, definen el rango de valores posibles del aesthetic y establecen la correspondencia entre éstos y los valores de la variable del set de datos que se le ha asignado.

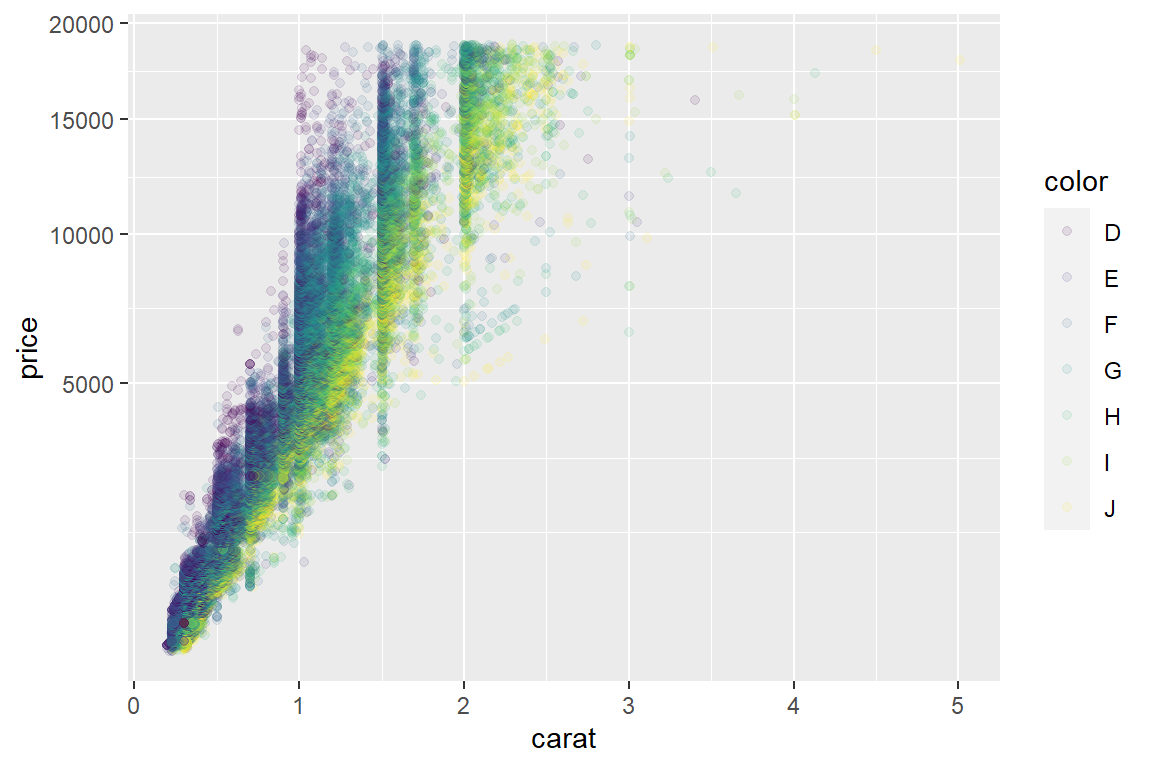
* Discretas vs continuas.
* De posición.
* De tamaño.
* De color.
* …

ggplot(data = diamonds, aes(x = carat, y = price, col = color)) +   
 geom\_point(alpha = 1/10)



## 6. Escalas (II)

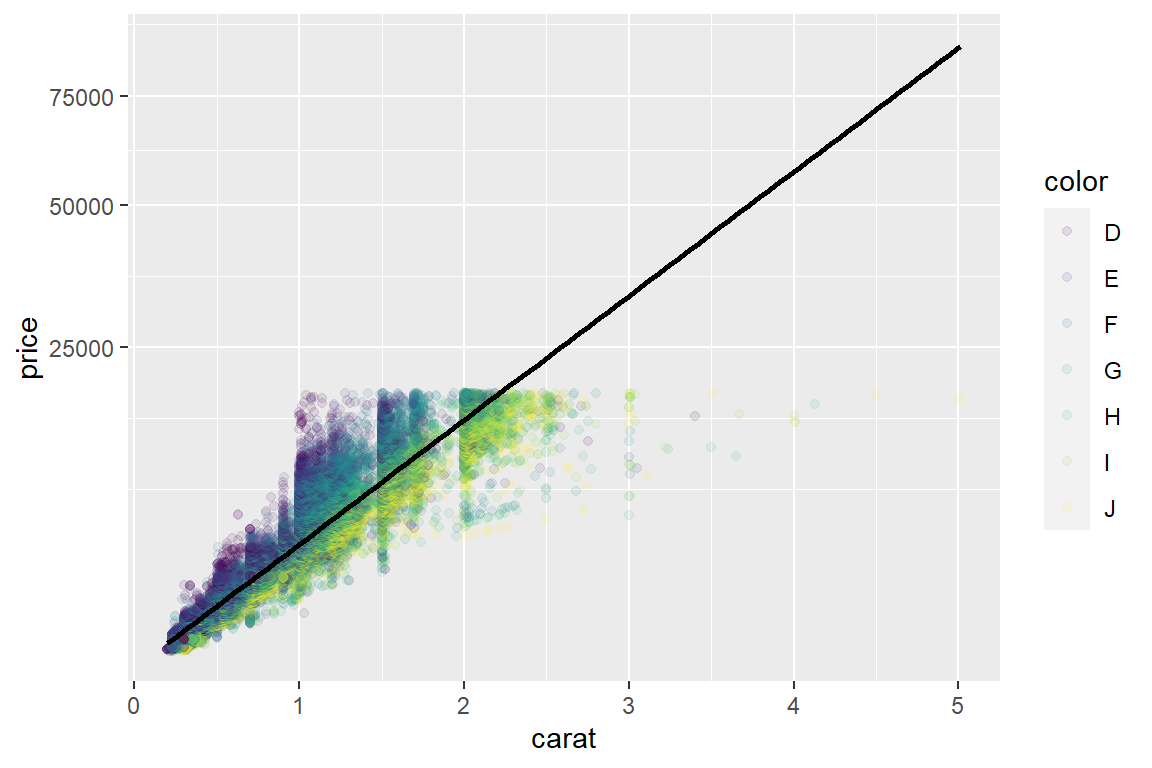
ggplot(data = diamonds, aes(x = carat, y = price, col = color)) +   
 geom\_point(alpha = 1/10) +  
 scale\_y\_sqrt()



## 6. Escalas (III)

ggplot(data = diamonds, aes(x = carat, y = price, col = color)) +   
 geom\_point(alpha = 1/10) +  
 scale\_y\_sqrt() +  
 geom\_smooth(method = 'lm', col = 'black')

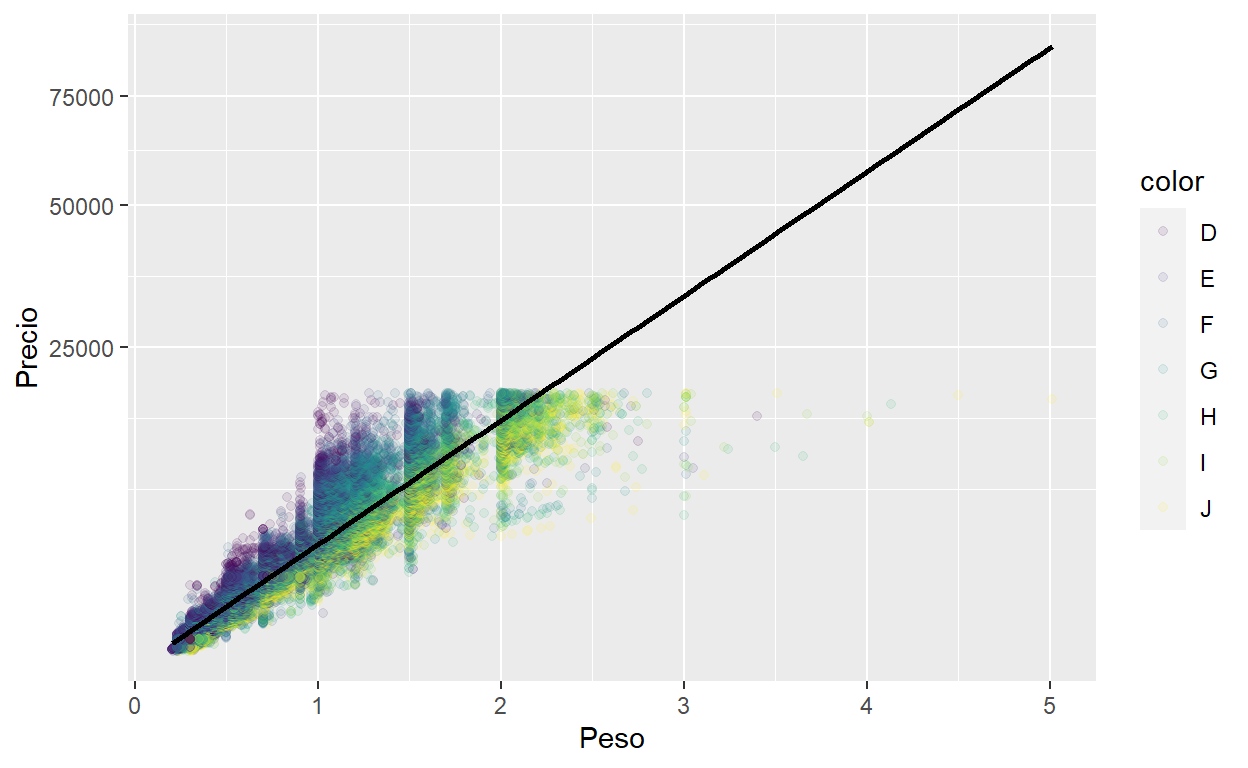
## `geom\_smooth()` using formula 'y ~ x'



## 6. Escalas (IV)

ggplot(data = diamonds, aes(x = carat, y = price, col = color)) +   
 geom\_point(alpha = 1/10) +  
 scale\_x\_continuous(name = 'Peso') +  
 scale\_y\_sqrt(name = 'Precio') +  
 geom\_smooth(method = 'lm', col = 'black')

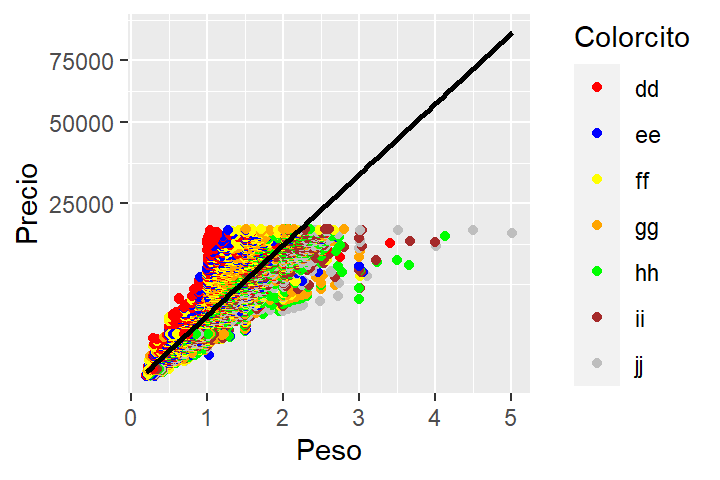
## `geom\_smooth()` using formula 'y ~ x'



## 6. Escalas (V)

ggplot(data = diamonds, aes(x = carat, y = price, col = color)) +   
 geom\_point() +  
 scale\_x\_continuous(name = 'Peso') +  
 scale\_y\_sqrt(name = 'Precio') +  
 geom\_smooth(method = 'lm', col = 'black') +  
 scale\_color\_manual(  
 name = 'Colorcito',  
 breaks = c('D', 'E', 'F', 'G', 'H', 'I', 'J'),  
 labels = c('dd', 'ee', 'ff', 'gg', 'hh', 'ii', 'jj'),  
 values = c('red', 'blue', 'yellow', 'orange', 'green', 'brown', 'gray')  
 )

## `geom\_smooth()` using formula 'y ~ x'



## 7. Temas (I)

Los temas (*themes*) sirven para modificar los aspectos visuales “no relativos a los datos” de un gráfico, tales como:

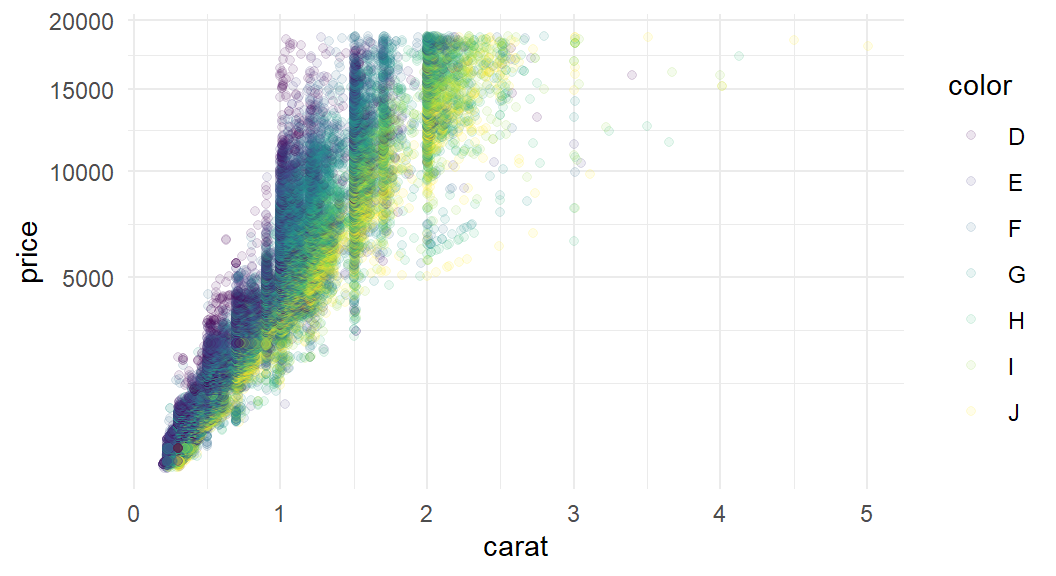
* Color y mallado del fondo.
* Posición de la leyenda
* Formato de los *ticks* de los ejes
* …

Existen tantas opciones que no tiene mucho sentido revisarlas todas, aunque pueden consultarse en la [documentación](http://ggplot2.tidyverse.org/reference/theme.html).

## 7. Temas (II)

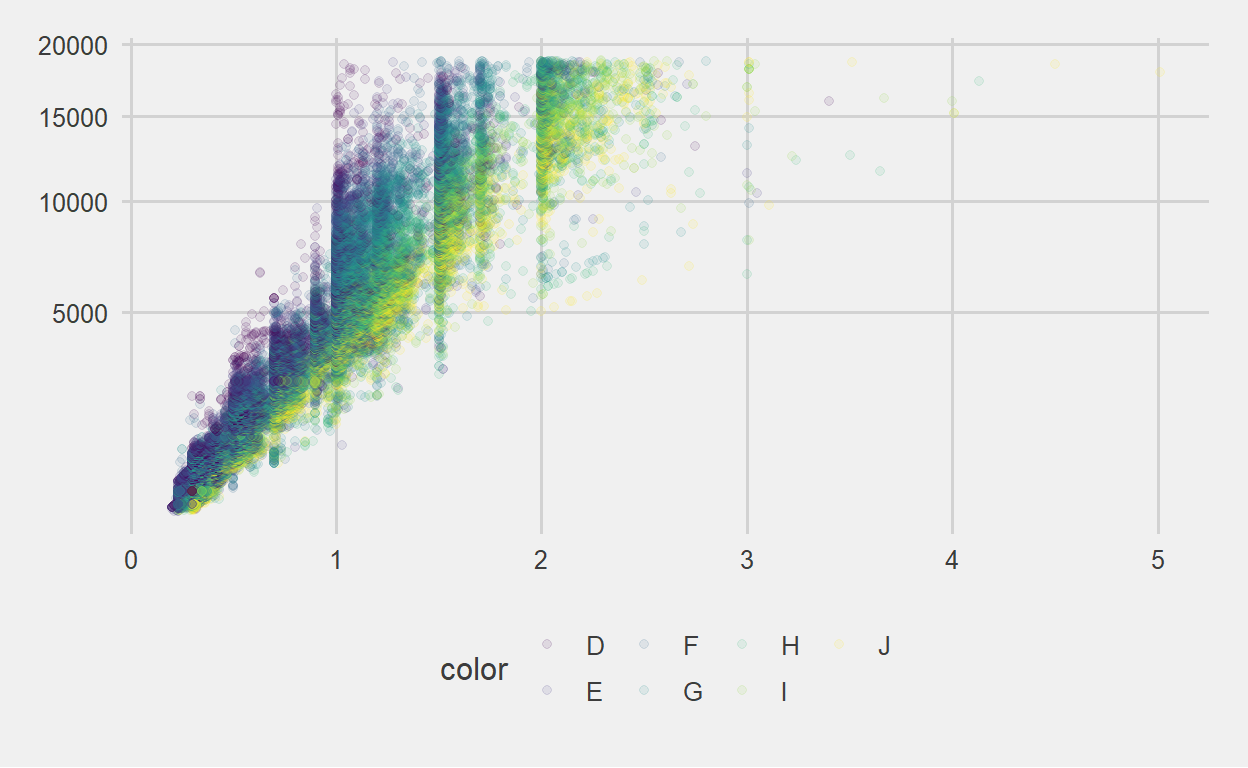
Muchas veces, con utilizar un tema predefinido para modificar el gráfico es más que suficiente.

ggplot(data = diamonds, aes(x = carat, y = price, col = color)) +   
 geom\_point(alpha = 1/10) +  
 scale\_y\_sqrt() +  
 theme\_minimal()



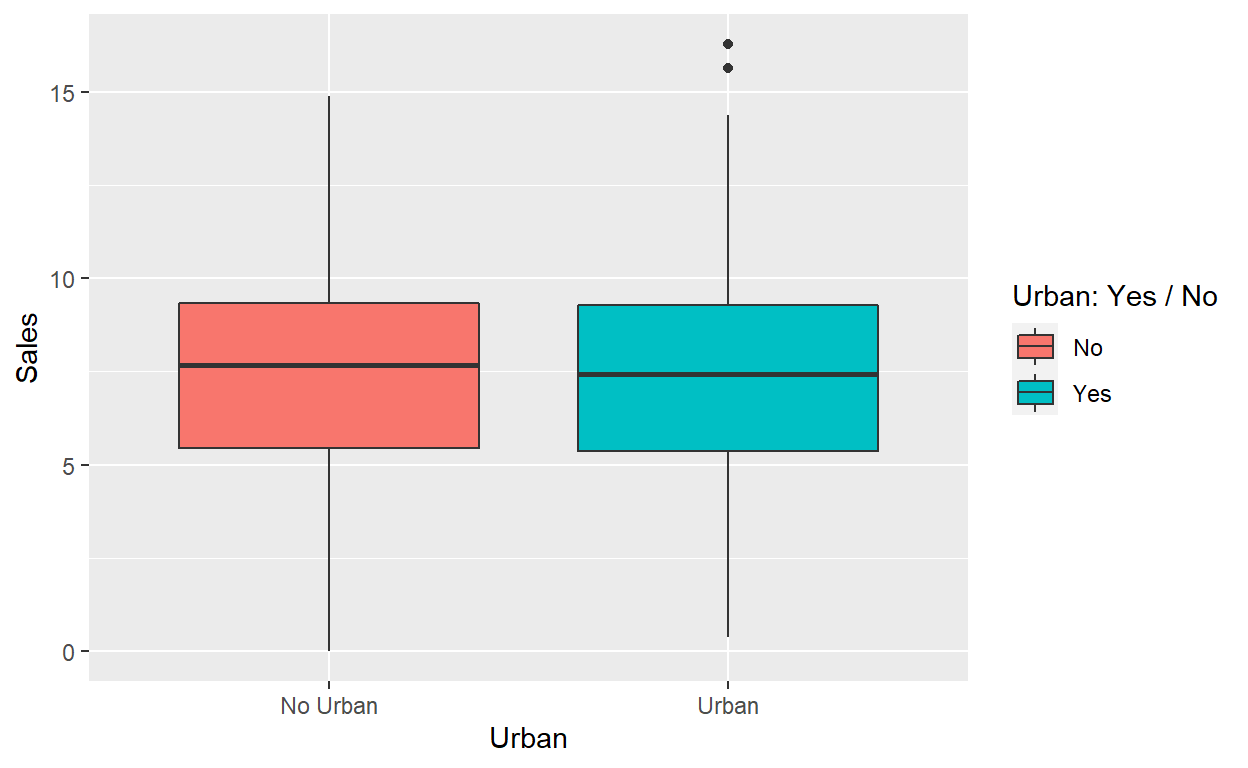
## 7. Temas (III)

#install.packages('ggthemes', dependencies=TRUE, repos='http://cran.rstudio.com/')  
library(ggthemes)  
  
ggplot(data = diamonds, aes(x = carat, y = price, col = color)) +   
 geom\_point(alpha = 1/10) +  
 scale\_y\_sqrt() +  
 theme\_fivethirtyeight()

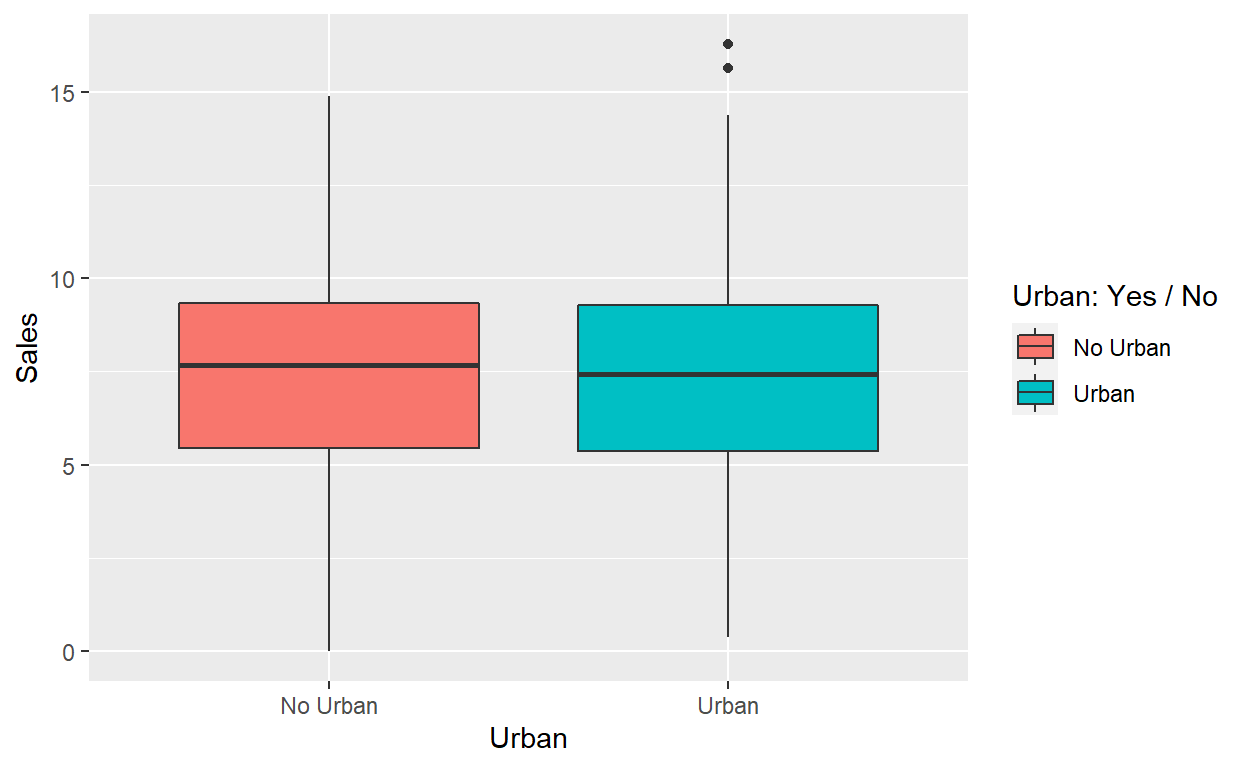


## 8. Leyendas

ggplot(data = Carseats, aes(x = Urban, y = Sales, fill = Urban)) +   
 geom\_boxplot() +   
 scale\_x\_discrete( breaks = c('No','Yes'), labels =c('No Urban','Urban') ) +   
 scale\_fill\_discrete(name = 'Urban: Yes / No')

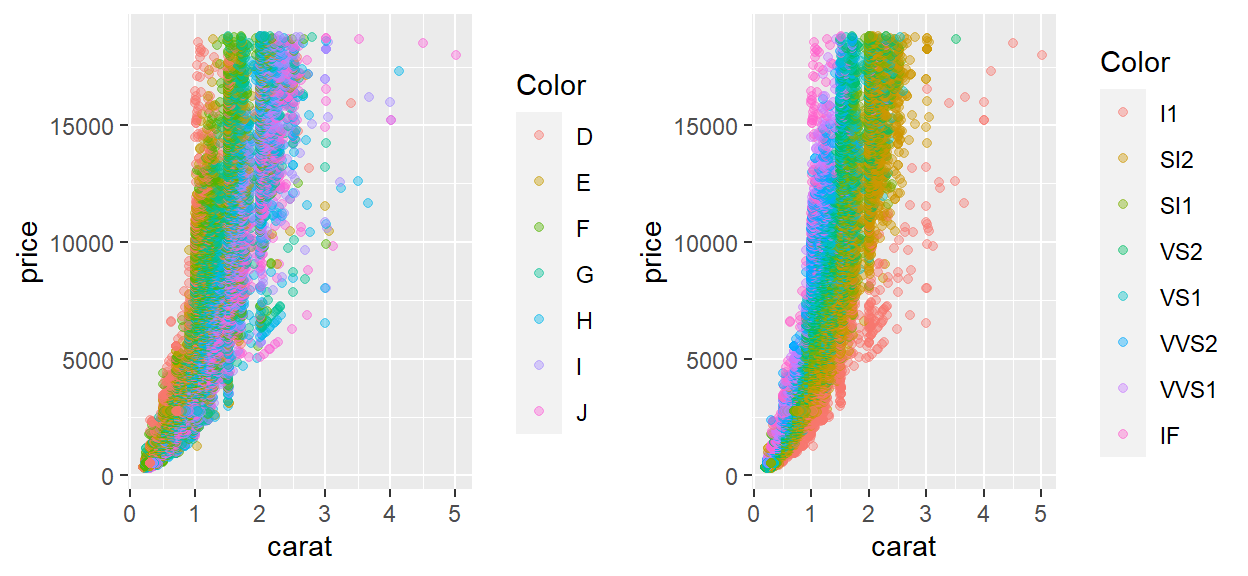


ggplot(data = Carseats, aes(x = Urban, y = Sales, fill = Urban)) +   
 geom\_boxplot() +   
 scale\_x\_discrete( breaks = c('No','Yes'), labels =c('No Urban','Urban') ) +   
 scale\_fill\_discrete(name = 'Urban: Yes / No', breaks = c('No','Yes'), labels =c('No Urban','Urban') )



## 9. Múltiples gráficos

library(gridExtra)  
g1 <- ggplot(data = diamonds, aes(x = carat, y = price, col = color)) +   
 geom\_point(alpha =0.4) +   
 scale\_color\_discrete( name = "Color" )  
g2 <- ggplot(data = diamonds, aes(x = carat, y = price, col = clarity)) +   
 geom\_point(alpha = 0.4) +   
 scale\_color\_discrete( name = "Color" )  
  
grid.arrange( g1, g2, nrow = 1)



## 10. Ejemplo más complejo

<http://rstudio-pubs-static.s3.amazonaws.com/284329_c7e660636fec4a42a09eed968dc47f32.html>