Capítulo 3

Gramática de gráficos

Librerías a utilizar:

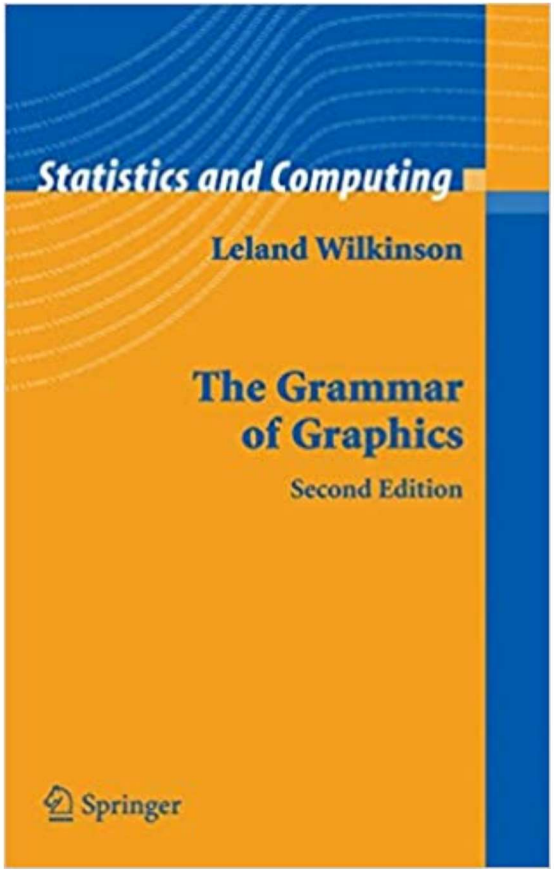
```
library(tidyverse)
library(datos)
library(patchwork) #sugerencia
```

```
str(millas)
```

```
## tibble [234 x 11] (S3: tbl_df/tbl/data.frame)
##  $ fabricante : chr [1:234] "audi" "audi" "audi" "audi" ...
##  $ modelo     : chr [1:234] "a4" "a4" "a4" "a4" ...
##  $ cilindrada : num [1:234] 1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...
##  $ anio       : int [1:234] 1999 1999 2008 2008 1999 1999 2008 1999 1999 2008 ...
##  $ cilindros  : int [1:234] 4 4 4 4 6 6 6 4 4 4 ...
##  $ transmision: chr [1:234] "auto(l5)" "manual(m5)" "manual(m6)" "auto(av)" ...
##  $ traccion   : chr [1:234] "d" "d" "d" "d" ...
##  $ ciudad     : int [1:234] 18 21 20 21 16 18 18 18 16 20 ...
##  $ autopista  : int [1:234] 29 29 31 30 26 26 27 26 25 28 ...
##  $ combustible: chr [1:234] "p" "p" "p" "p" ...
##  $ clase      : chr [1:234] "compacto" "compacto" "compacto" "compacto" ...
```

Gramática de gráficos

La gramática de gráficos (Wilkinson, 2005) es una deconstrucción en primeros principios de las gráficas. Nos brinda una serie de reglas y atributos (gramática) que nos permite ir mas allá de las gráficas comunes (palabras) a gráficas complejas (frases).

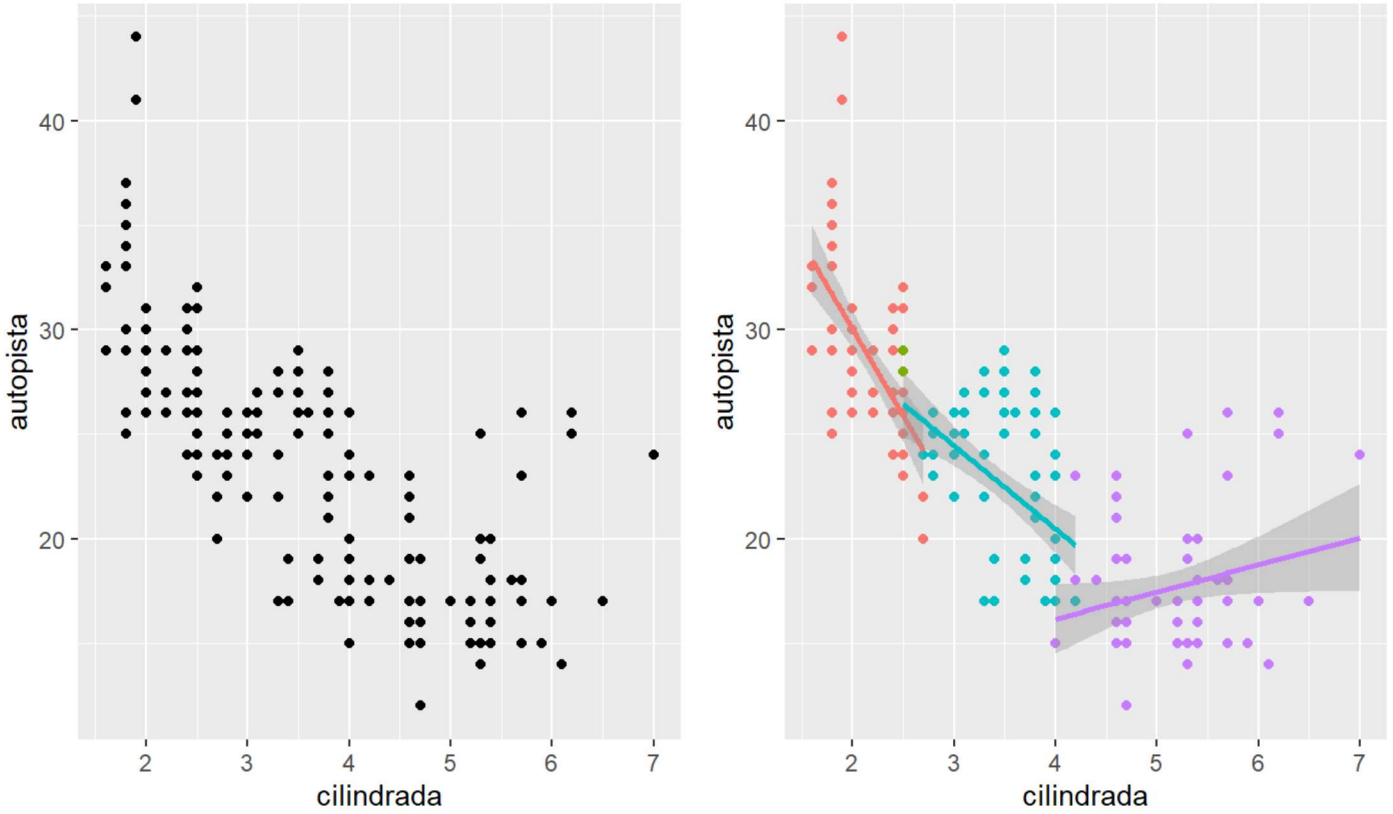


Se utiliza en varias librerias:

- ggplot2
- Tableau
- Vega-Lite
- Plotly
- Altair

Ejemplo:

A partir de gráficas sencillas (scatterplot) podemos construir gráficas complejas agregando capas.



¿Cómo funciona?

La implementación de la gramática en ggplot2 se hace mediante capas.

```
ggplot() +
  layer(
    data = millas,
    mapping = aes(cilindrada, autopista),
    geom = "point",
    stat = "identity",
    position = "identity"
  )
```

El arquetipo que vamos a usar:

```
ggplot(data= <DATOS>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPEOS>))
```

Componentes:

- **data:**
  - Los datos a usar. Precisan estar en formato *tidy*.
- **mapping:**
  - Un conjunto de asignaciones estéticas. Especficiadas mediante el comando **aes()**. Describen cómo las variables se transforman en propiedades visuales que percibimos en las gráficas. Por ejemplo, las variables asignadas en los ejes, los colores, el tamaño, la forma, etc.
- **geom:**
  - Los objetos geométricos (**geom_**) renderizan la capa, controlando que tipo de gráfica resulta.
- **stat:**
  - Son transformaciones de los datos, generalmente creadas por las funciones **geom_**. En algunos casos se computan manualmente (ej: **stat_ecdf()**).
- **position:**
  - Ajusta la posición de los elementos en la capa. En el caso de solapamiento se puede definir la lógica para resolver el conflicto mediante el uso de este ajuste. (ej. **position_jitter()** para puntos, **position_dodge()** para histogramas.)

Geometrias comunes:

Primitivas graficas comunes:

- **geom_blank():** No muestra nada.
- **geom_point():** Puntos.
- **geom_path():** Caminos.
- **geom_line():** Caminos ordenados en x.
- **geom_polygon():** Polígonos rellenos.
- **geom_abline()** y **geom_hline()** Lineas verticales y horizontales.
- **geom_text():** Texto.

En una variable:

- Discreta:
  - **geom_bar():** Distribución.
- Continua:
  - **geom_histogram():** Histograma.
  - **geom_density():** Estimación de densidad.

Dos variables:

- Ambas continuas:
  - **geom_smooth():** Ajuste continuo.
- Distribucion:
  - **geom_bin2d()** y **geom_hex():** Discretiza y cuenta.
  - **geom_density2d():** Densidad en 2d.
- Al menos una discreta:
  - **geom_jitter():** Agrega ruido a puntos.
- Una continua y una discreta:
  - **geom_boxplot():** Boxplots.
  - **geom_violin():** Violin plot.

Incertidumbre:

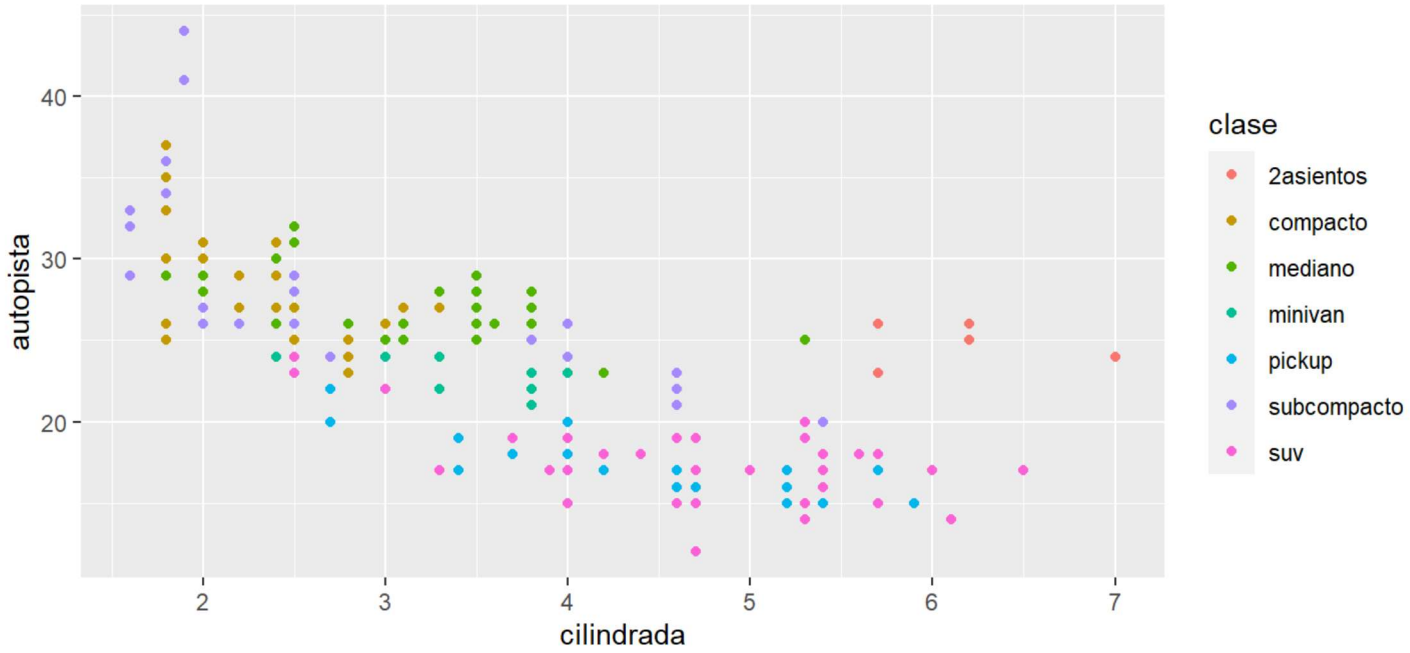
- **geom_errorbar()** Barras de errores.

- Muchas mas.

Ejemplos

Se asigna a la estética *color* la variable *clase* del dataframe. Ggplot2 mediante **scales** mapea la variable discreta *clase* a puntos equiespaciados en un plano de color (ej. HCL).

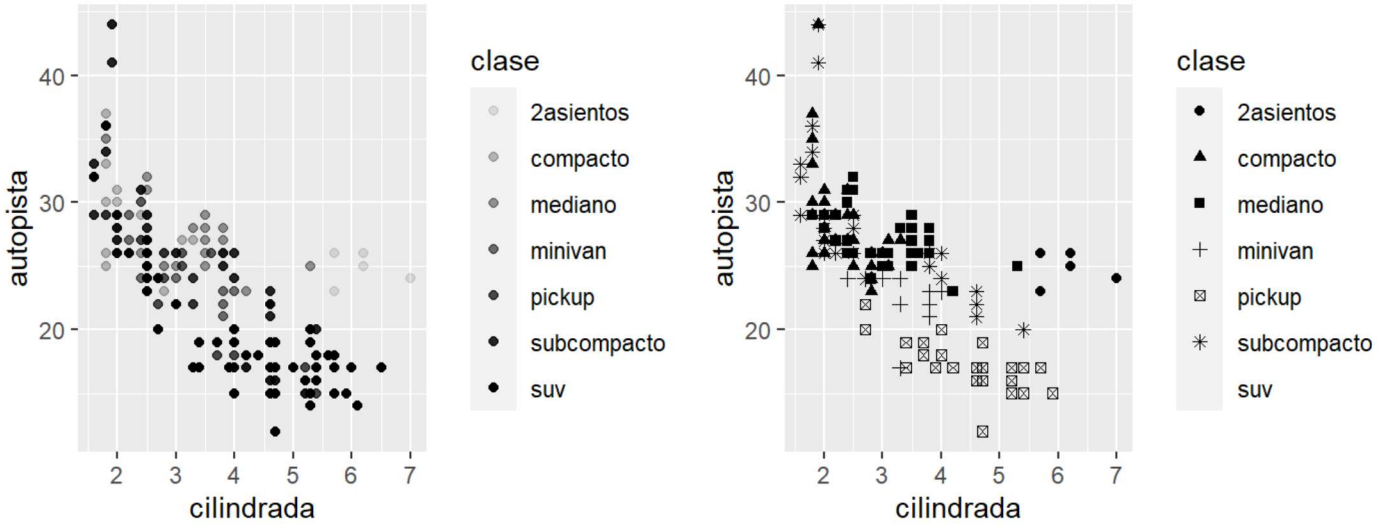
```
ggplot(data=millas)+
  geom_point(mapping=aes(x=cilindrada, y=autopista, color=clase))
```



Ejemplos

Podemos asignarle la transparencia o la forma a la columna clase.

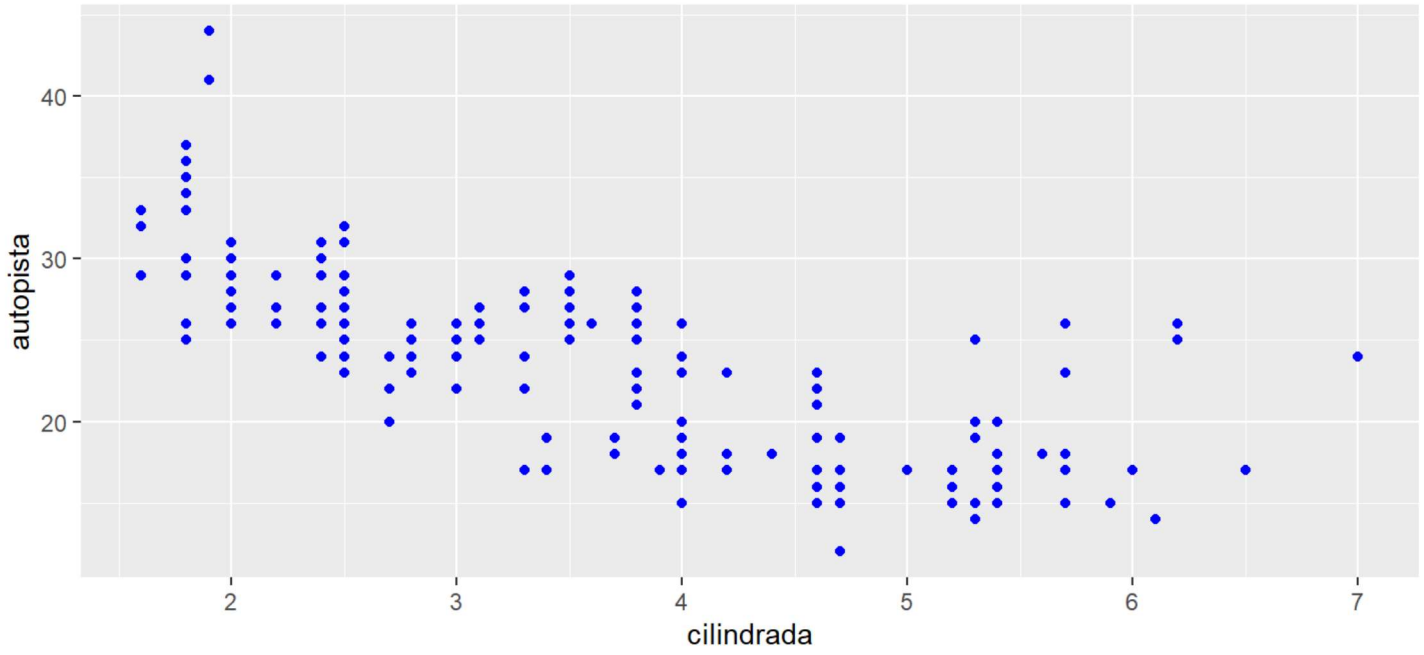
```
p1 <- ggplot(data=millas)+
        geom_point(aes(x=cilindrada, y=autopista, alpha=clase))
p2 <- ggplot(data=millas)+
      geom_point(aes(x=cilindrada, y=autopista, shape=clase))
p1 + p2
```



Ejemplos

Se pueden fijar las propiedades estéticas, asignando la variable fuera de **aes()**. Por ejemplo fijamos el color:

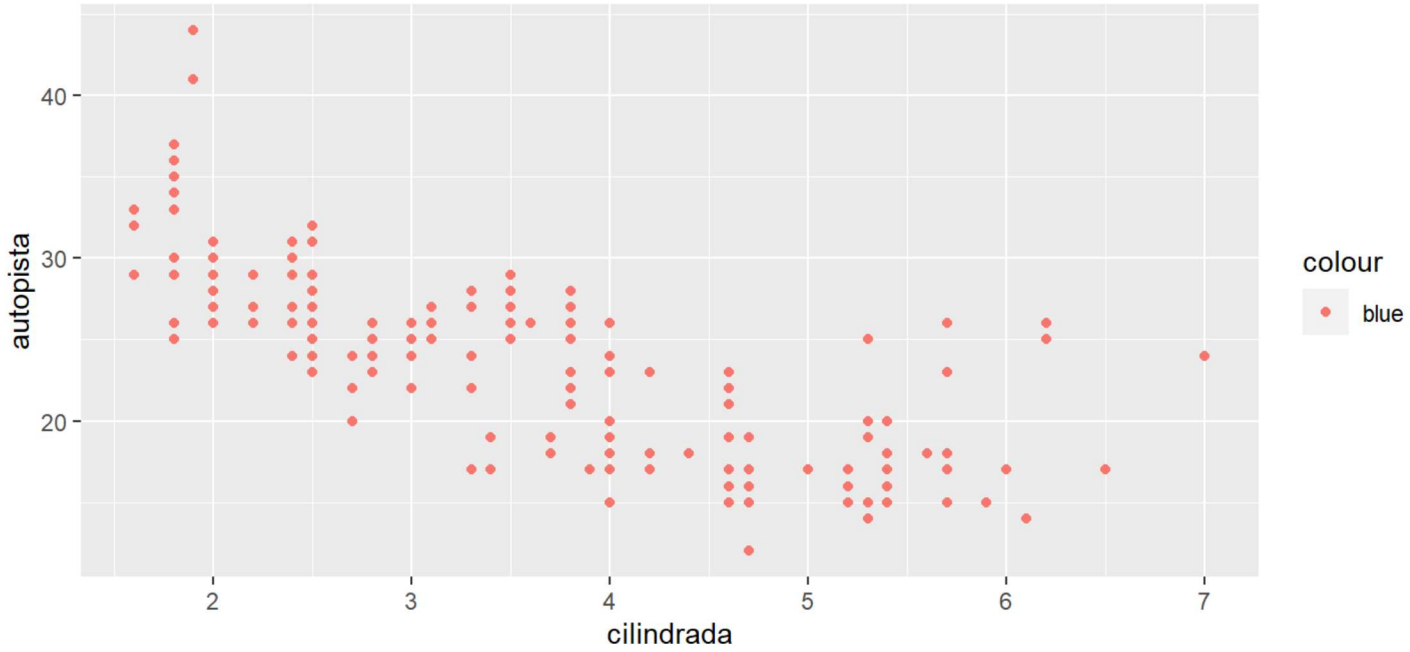
```
ggplot(data=millas)+
  geom_point(mapping=aes(x=cilindrada, y=autopista), color='blue')
```



Ejemplos

¿Qué pasa si fijamos la variable estética dentro de **aes()**? Para cada entrada en la gráfica (x,y) se crea una variable color y se le asigna la magnitud 'blue'. Luego, ggplot2 mediante **scales** elige un punto en el plano HLC y se lo asigna al string 'blue'. Este punto es el color rojo/naranja observado.

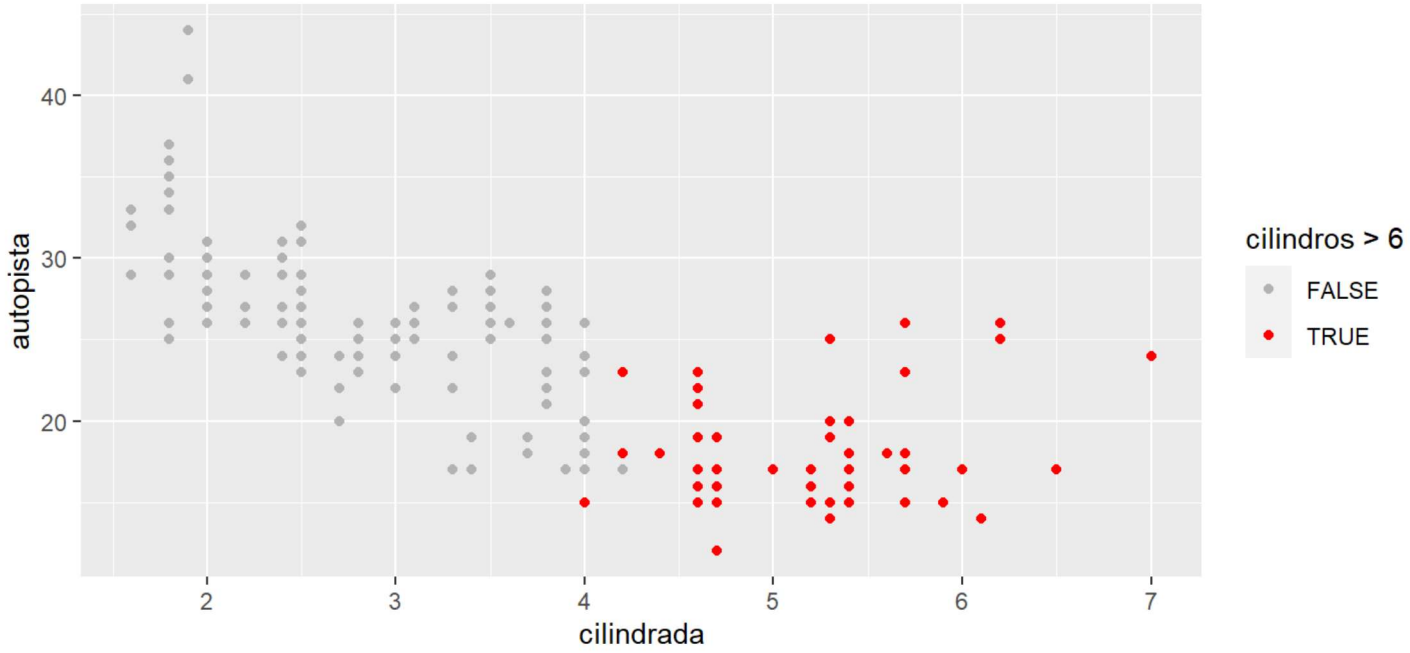
```
ggplot(data=millas)+
  geom_point(mapping=aes(x=cilindrada, y=autopista, color='blue'))
```



Ejemplos

Se pueden utilizar expresiones en las variables estéticas.

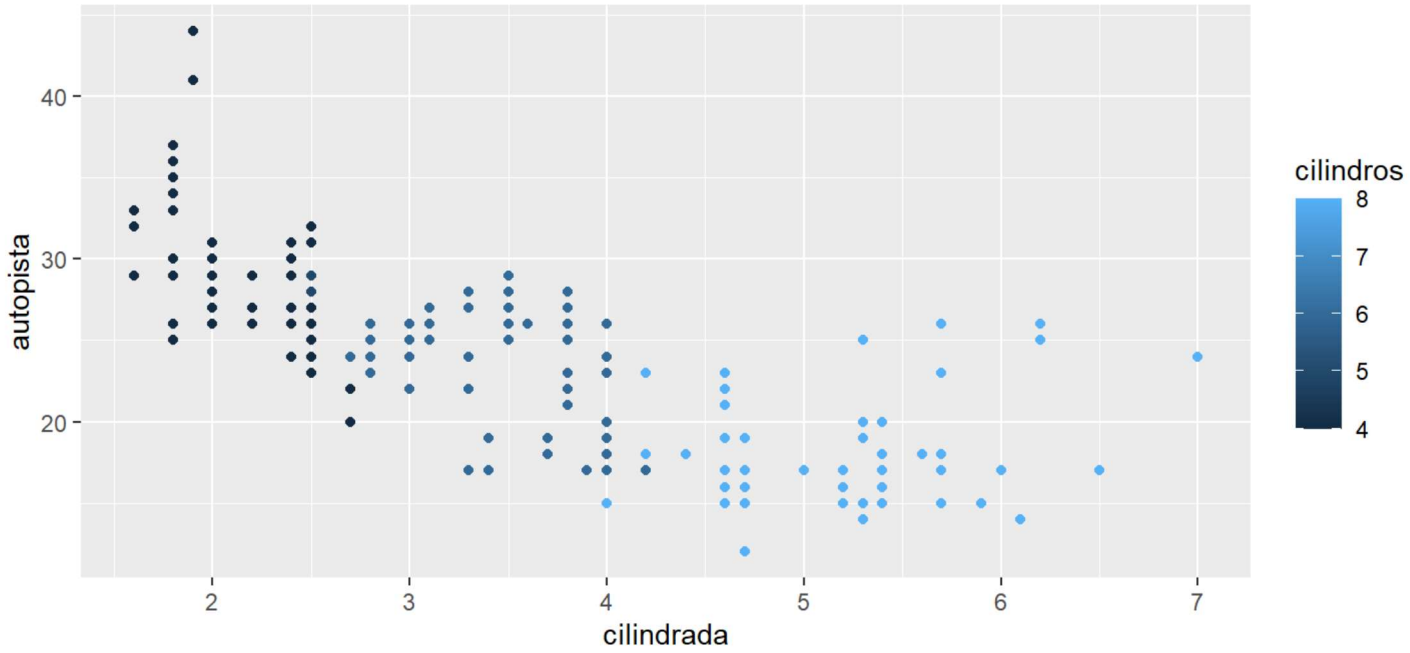
```
ggplot(millas, aes(x=cilindrada, y=autopista)) +
  geom_point(aes(color= cilindros > 6))+
  scale_color_manual(values=c('grey70','red'))
```



Ejemplos

Se puede asignar colores a una variable continua. En este caso ggplot2 traza una curva en el espacio de colores HLC para los valores de la variable.

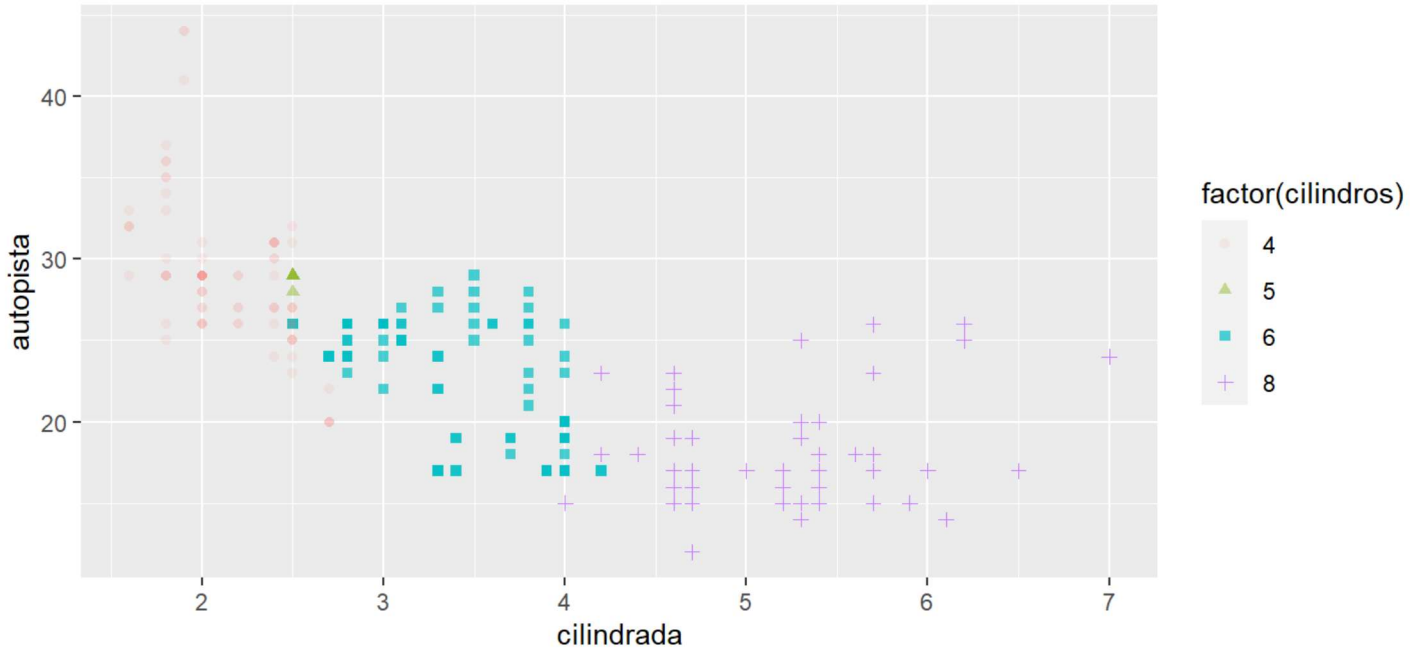
```
ggplot(millas, aes(x=cilindrada, y=autopista)) +
  geom_point(aes(color=cilindros))
```



Ejemplos

Podemos incluso asignar la misma variable a múltiples estéticas.

```
ggplot(millas, aes(x=cilindrada, y=autopista)) +
  geom_point(aes(color=factor(cilindros),
                 shape=factor(cilindros),
                 alpha=factor(cilindros)))
```

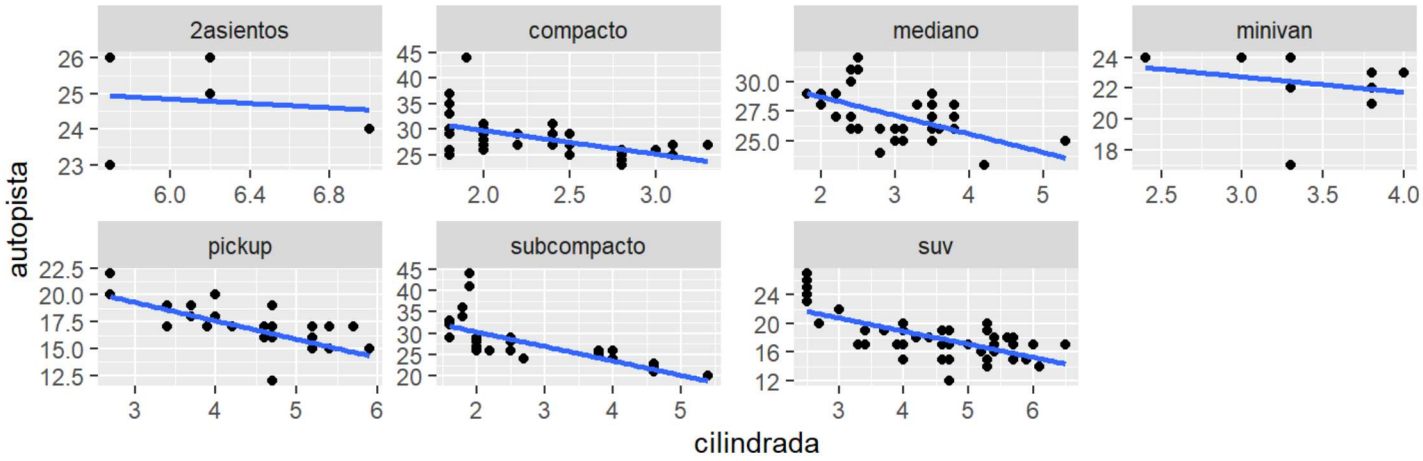


Facetas

Es un mecanismo para mostrar múltiples gráficos en una misma pagina. Divide los datos en subconjuntos y grafica éstos en cada panel. Hay dos modalidades: **facet_wrap()** y **facet_grid()**.

Para **facet_wrap()** el primer argumento debe ser una variable categórica. Se especifica mediante ~ . El parámetro **scales** fija o libera los ejes para que sean compartidos o no.

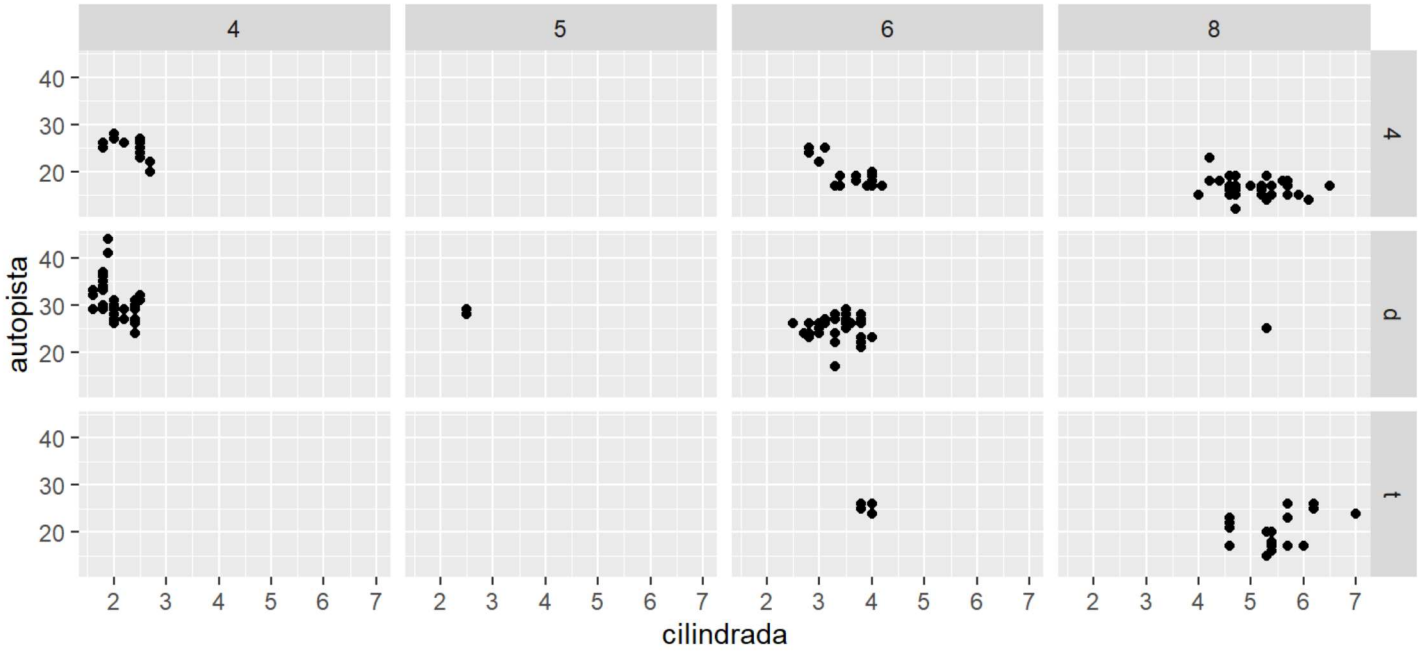
```
ggplot(millas, aes(x=cilindrada, y=autopista)) +
  geom_point() +
  geom_smooth(method='lm', se=F) +
  facet_wrap(~clase, nrow=2, scales="free")
```



Facetas

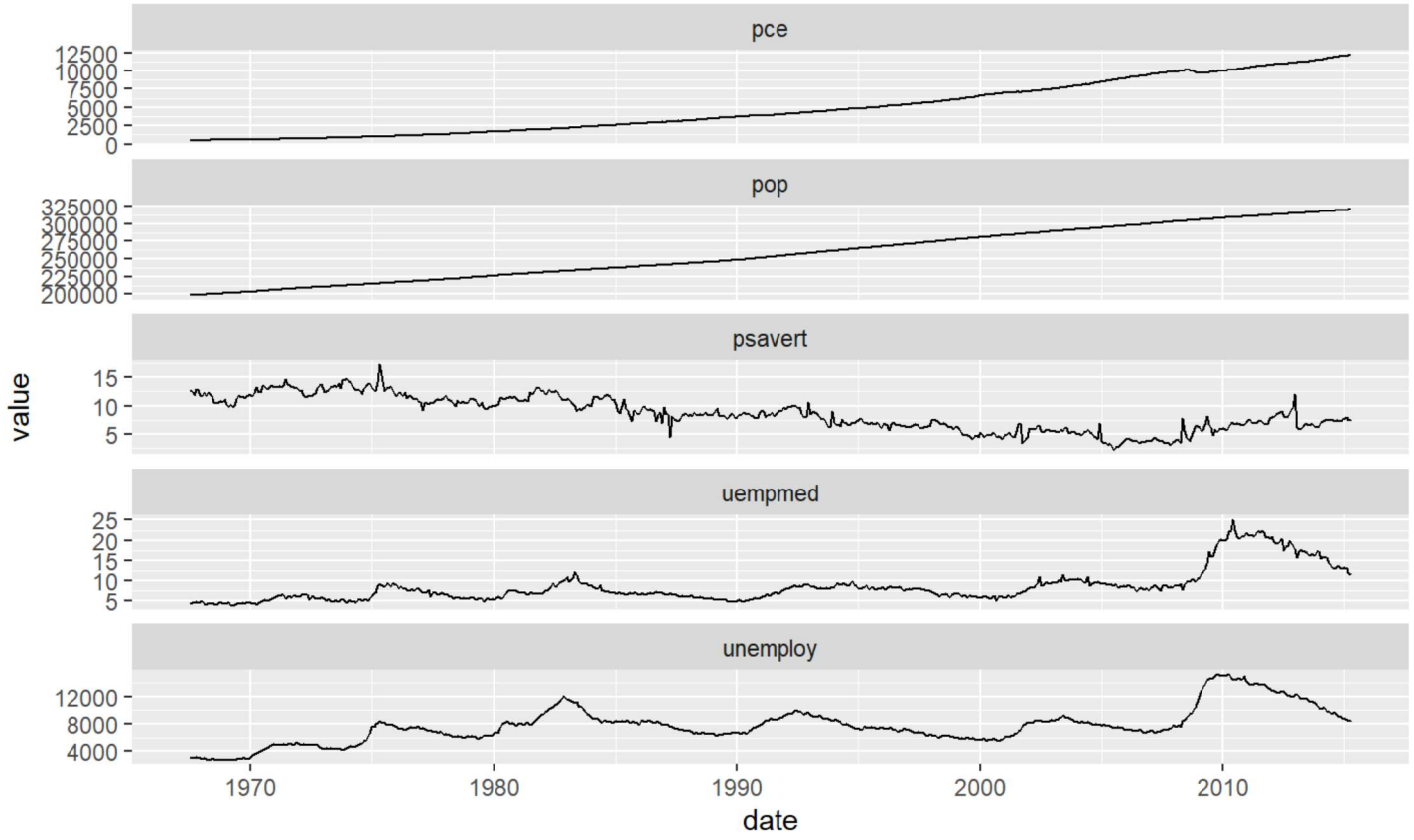
facet_grid() separa los datos en combinaciones de a dos variables especificadas por a ~ b. Si se quiere suprimir una de las variables, se sustituye por un punto ., por ejemplo: (. ~ cilindros).

```
ggplot(millas, aes(x=cilindrada, y=autopista)) +
  geom_point() +
  facet_grid(traccion ~ cilindros)
```



Ejemplo

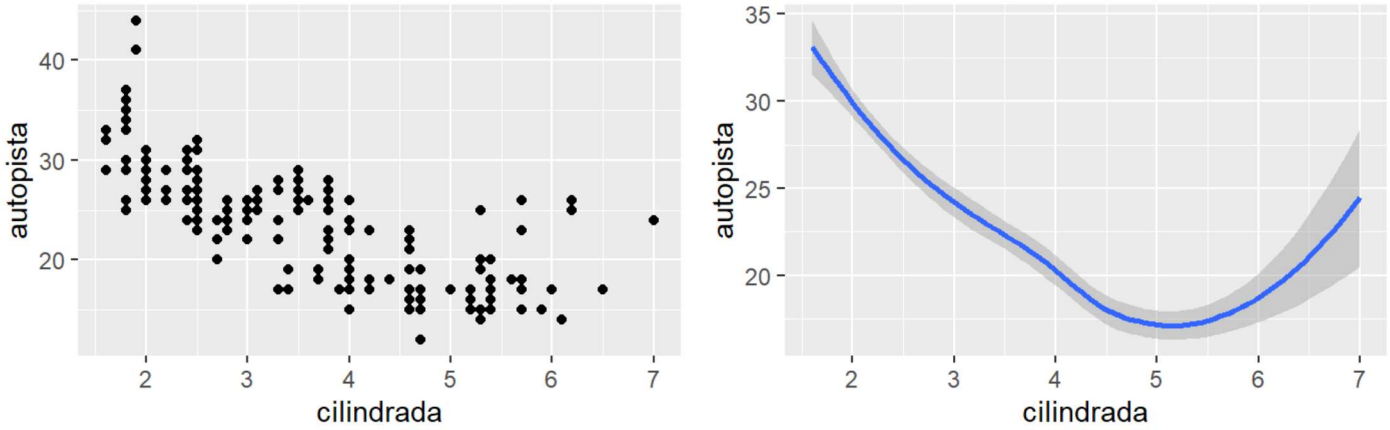
```
ggplot(economics_long, aes(date, value)) +
  geom_line() +
  facet_wrap(~variable, scales="free_y", ncol=1)
```



Objetos geometricos

Comparemos los siguientes gráficos. Tiene geometrías distintas. La representación gráfica de los puntos viene dada por el objecto geométrico que se utiliza.

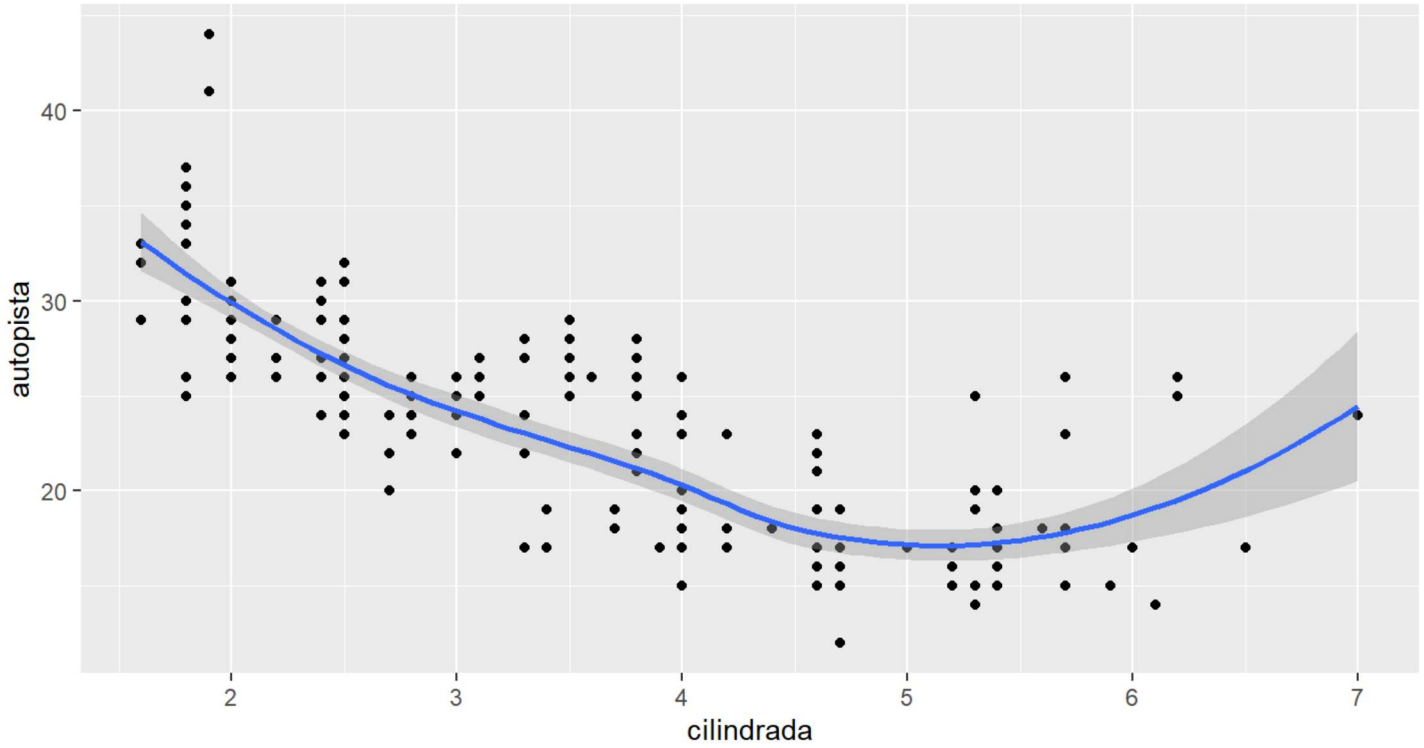
```
p1 <- ggplot(millas, aes(cilindrada, autopista)) +
        geom_point()
p2 <- ggplot(millas, aes(cilindrada, autopista)) +
        geom_smooth()
p1 + p2
```



Ejemplo

Como la gramática es representada en capas, para agregar varias simplemente se usa (+).

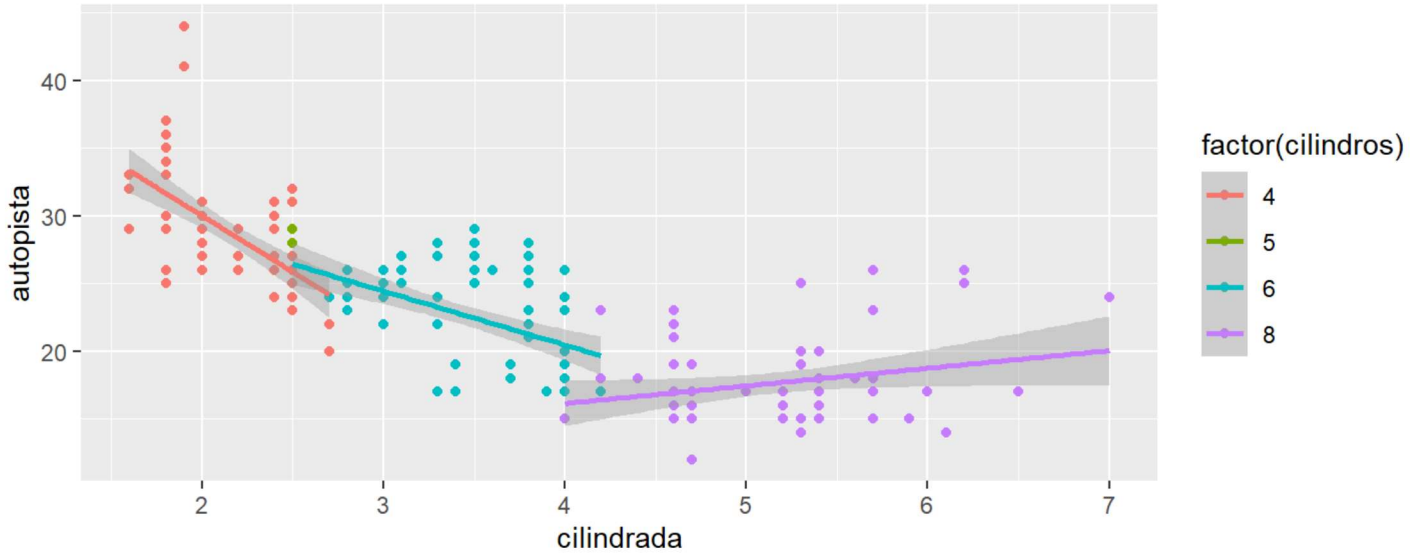
```
ggplot(millas, aes(cilindrada, autopista)) +
  geom_point() +
  geom_smooth()
```



Ejemplo

Podemos incluso superponerlas y generar el gráfico que teníamos al principio. Donde para cada valor de cilindros diferentes se le superpone una curva de regresión lineal. Las geometrías heredan **color** de la definición global de ggplot.

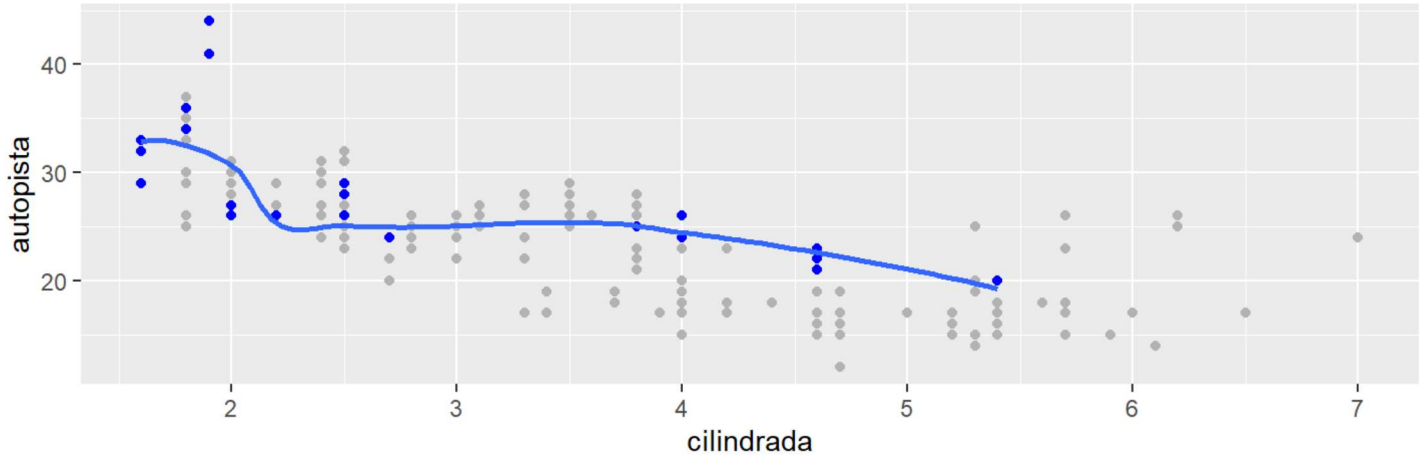
```
ggplot(millas, aes(cilindrada, autopista, color = factor(cilindros))) +
  geom_point() +
  geom_smooth(method = 'lm')
```



Ejemplo

Los parámetros dentro de las geometrías sobreescriben a los parámetros globales. Si se quiere suprimir algún parámetro global se utiliza **NULL**. Por ejemplo, en la siguiente gráfica se ajusta por regresión local los valores correspondientes a la clase "subcompacto".

```
ggplot(millas, aes(cilindrada, autopista)) +
  geom_point(aes(color=clase == "subcompacto")) +
  scale_color_manual(values=c('grey70','blue')) +
  geom_smooth(data=filter(millas, clase=="subcompacto"), se=FALSE) +
  theme(legend.position = "none")
```

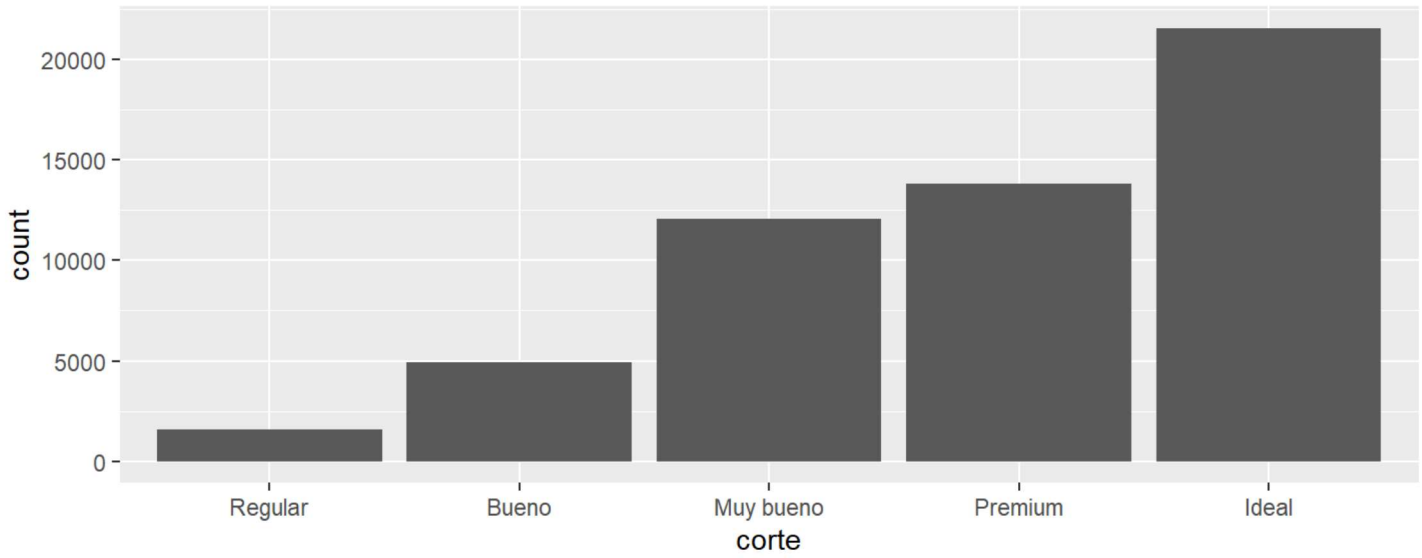


Transformaciones estadísticas

Algunas geometrías implementan transformaciones estadísticas de los datos para generar gráficas. Como por ejemplo los gráficos de barras. En este ejemplo la variable **count** no esta definida en los datos, la genera la geometría.

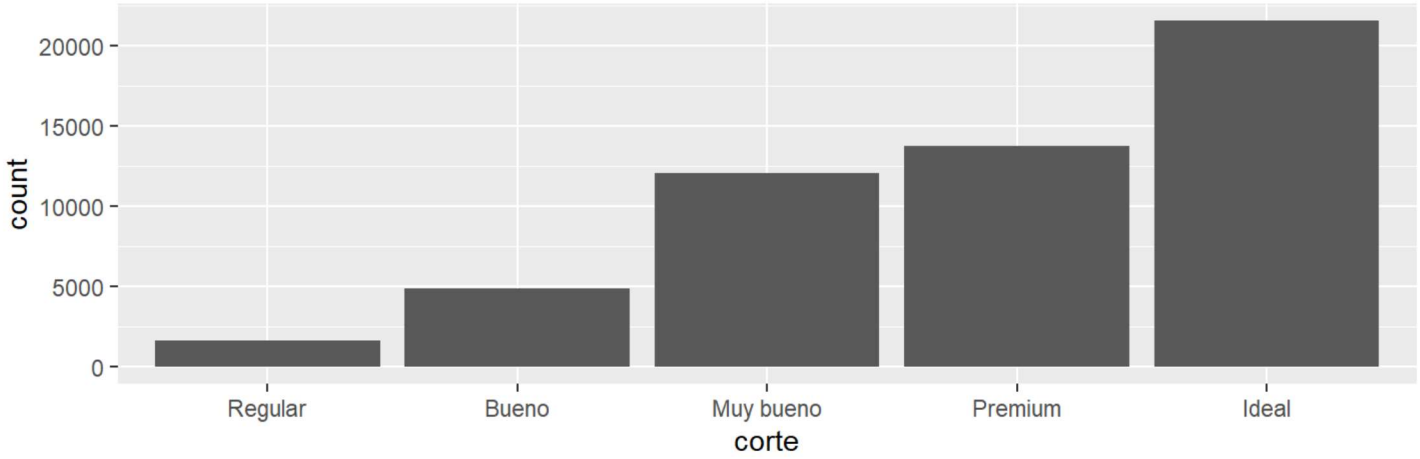
Otros ejemplos son las geometrias **geom_smooth()** que calcula parámetros de regresión local o lineal, o **geom_boxplot()** que calcula un resumen estadístico de las variables.

```
ggplot(diamantes) +
  geom_bar(aes(x = corte))
```



La relación entre estadísticas y geometrías es reciproca. Cada estadística tiene una geometría predeterminada. Por ejemplo:

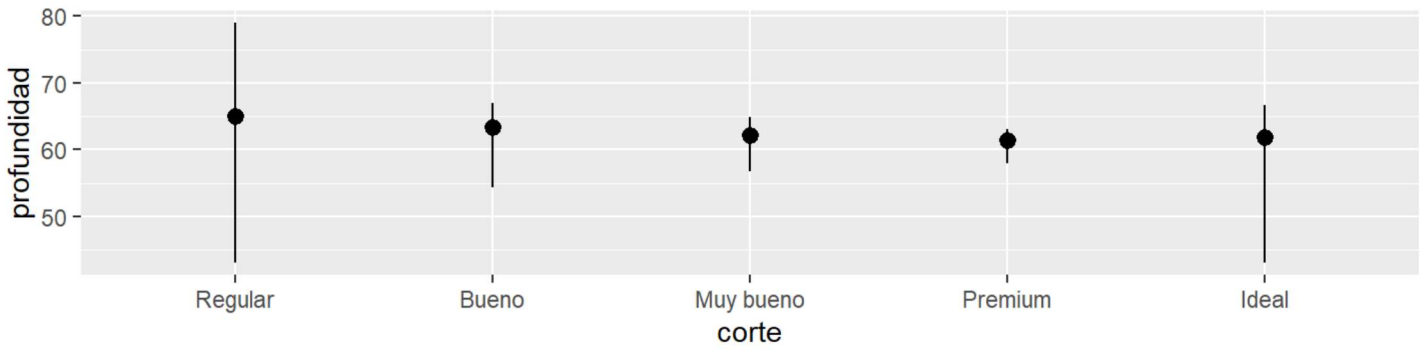
```
ggplot(diamantes) +
  stat_count(aes(x = corte))
```



Comúnmente se utilizan las funciones de geometría. Sin embargo hay algunas razones para especificar las funciones estadísticas.

- Anular la estadística predeterminada por una customizada.
- Cambiar el mapeo de las variables estéticas, por ejemplo en vez de conteo usar proporción.
- Si se quiere reslatar una transformación estadistíca en particular.
- Si la estadística que se quiere no tiene una función geométrica asociada.

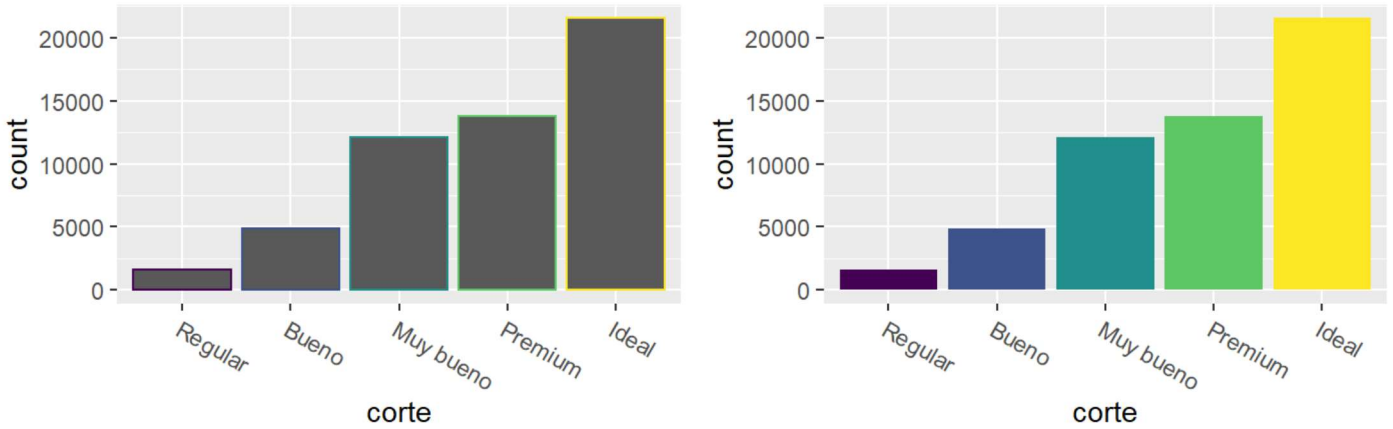
```
ggplot(diamantes) +
  stat_summary(
    mapping = aes(corte, profundidad),
    fun.min = min,
    fun.max = max,
    fun = median
  )
```



Ajustes de posición

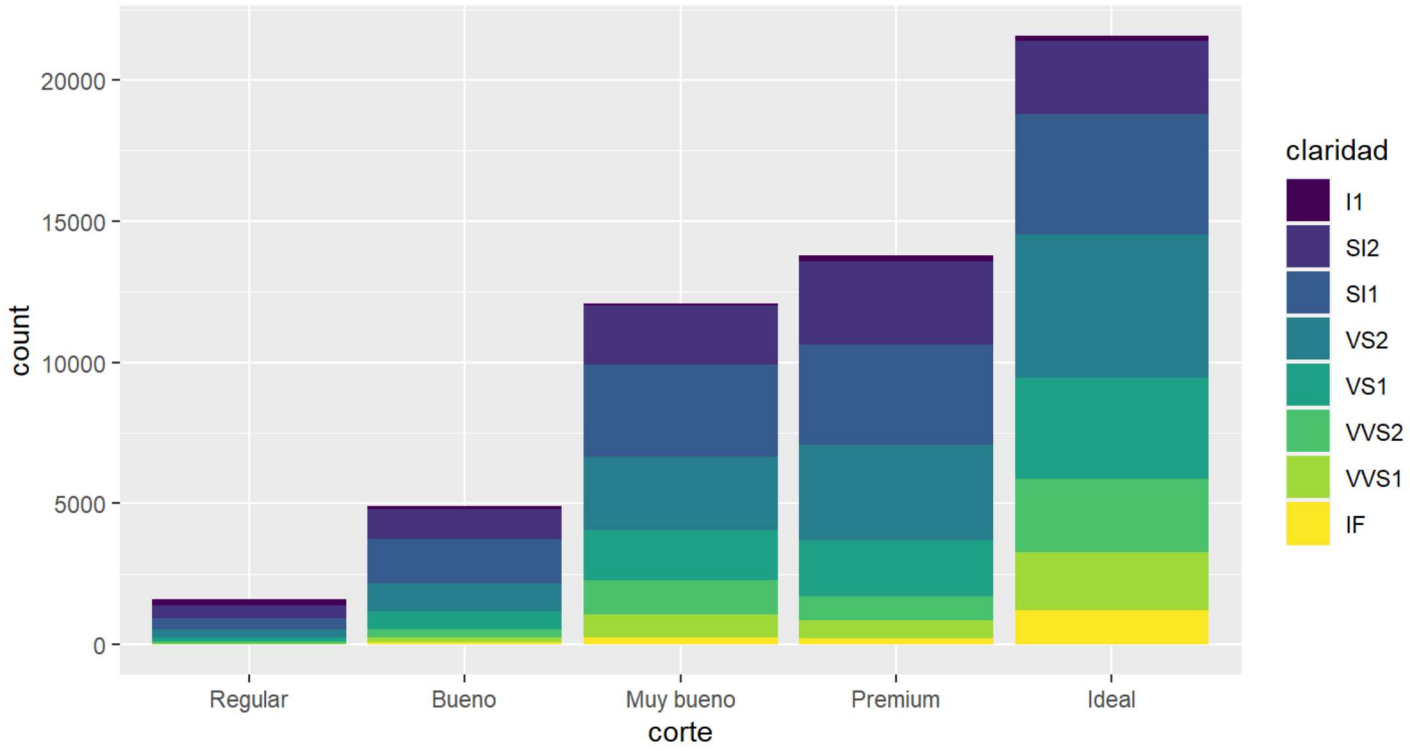
Podemos colorear los graficos de barras utilizando la estética **color** y **fill**.

```
p1 <- ggplot(diamantes, aes(x = corte)) +
        geom_bar(aes(color=corte)) +
        theme(legend.position = "none",
              axis.text.x = element_text(angle = -30, vjust= 1, hjust = 0))
p2 <- ggplot(diamantes, aes(x = corte)) +
        geom_bar(aes(fill=corte)) +
        theme(legend.position = "none",
              axis.text.x = element_text(angle = -30, vjust= 1, hjust = 0))
p1 + p2
```



Sin embargo si se utiliza otra variable para rellenar, las barras se apilan automáticamente. Donde cada rectángulo corresponde a una combinación de *corte* y *claridad*.

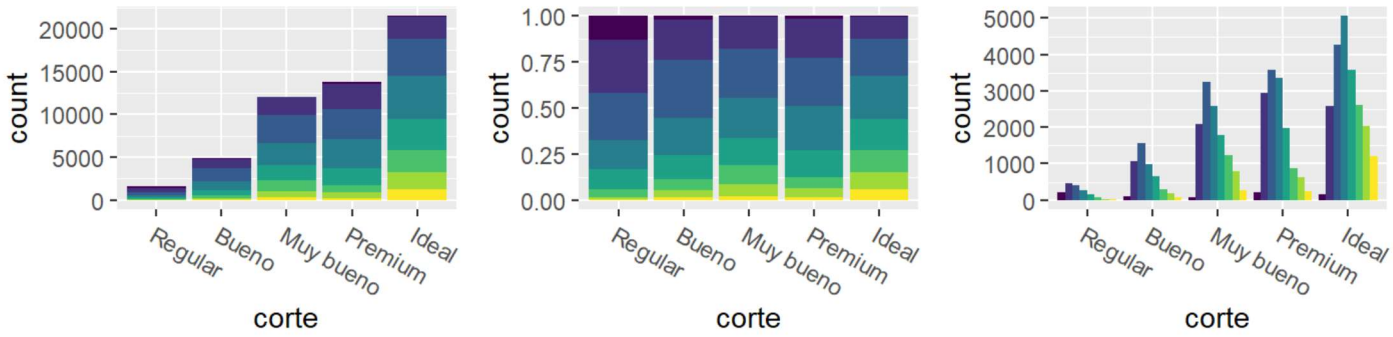
```
ggplot(data=diamantes) +
  geom_bar(mapping = aes(x=corte, fill=claridad))
```



La variable **position** nos da alternativas al gráfico anterior. Hay tres modalidades:

- **position_stack():** Apila las barras.
- **position_fill():** Apila las barras pero el tope siempre esta en 1.
- **position_dodge():** Coloca las barras lado a lado.

```
dplot <- ggplot(data=diamantes, aes(x=corte, fill=claridad)) +
          theme(legend.position = "none",
                axis.text.x = element_text(angle = -30, vjust= 1, hjust = 0))
p1 <- dplot + geom_bar()
p2 <- dplot + geom_bar(position="fill")
p3 <- dplot + geom_bar(position="dodge")
p1 + p2 + p3
```



Ejemplo

Cuando se trabaja con gráficos de puntos **geom_point()** hay tres modalides de posición.

- **position_nudge():** Mueve los puntos por un offset especificado.
- **position_jitter():** Agregua un ruido aleatorio a cada posición.
- **position_jitterdodge():** Esquiva puntos dentro de grupos y luego agrega ruido.

```
p1 <- ggplot(millas, aes(cilindrada, autopista, alpha=0.5))+
        geom_point() +
        theme(legend.position = "none")
p2 <- ggplot(millas, aes(cilindrada, autopista, alpha=0.5)) +
      geom_point(position=position_jitter(width=0.5, height=0.5)) +
      theme(legend.position = "none")
p1 + p2
```