

**ggplot2**

**Fundamentos lenguajes: R**

**Alberto Torres Barrán y Irene Rodríguez Luján**

**2019-07-08**

# ggplot2

## Introducción

- Implementa una gramática de gráficos en R
- Divide un gráfico en sus componentes esenciales
- Múltiples ventajas con respecto a los gráficos de R base
  - Leyenda automática
  - Facetas
  - ...

# Gramática de gráficos

- *mapping* se define con `aes()` (*aesthetics*) y describe como las variables de un data frame se asignan a propiedades visuales
- *data* data frame
- *geom* objetos geométricos con el que se van a representar los datos
- *stat* transforman los datos
- *position* pequeños ajustes en la posición de los elementos

# Ejemplo

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

# Plantilla

- El gráfico más sencillo consta como mínimo de los siguientes componentes [**Fuente**]:

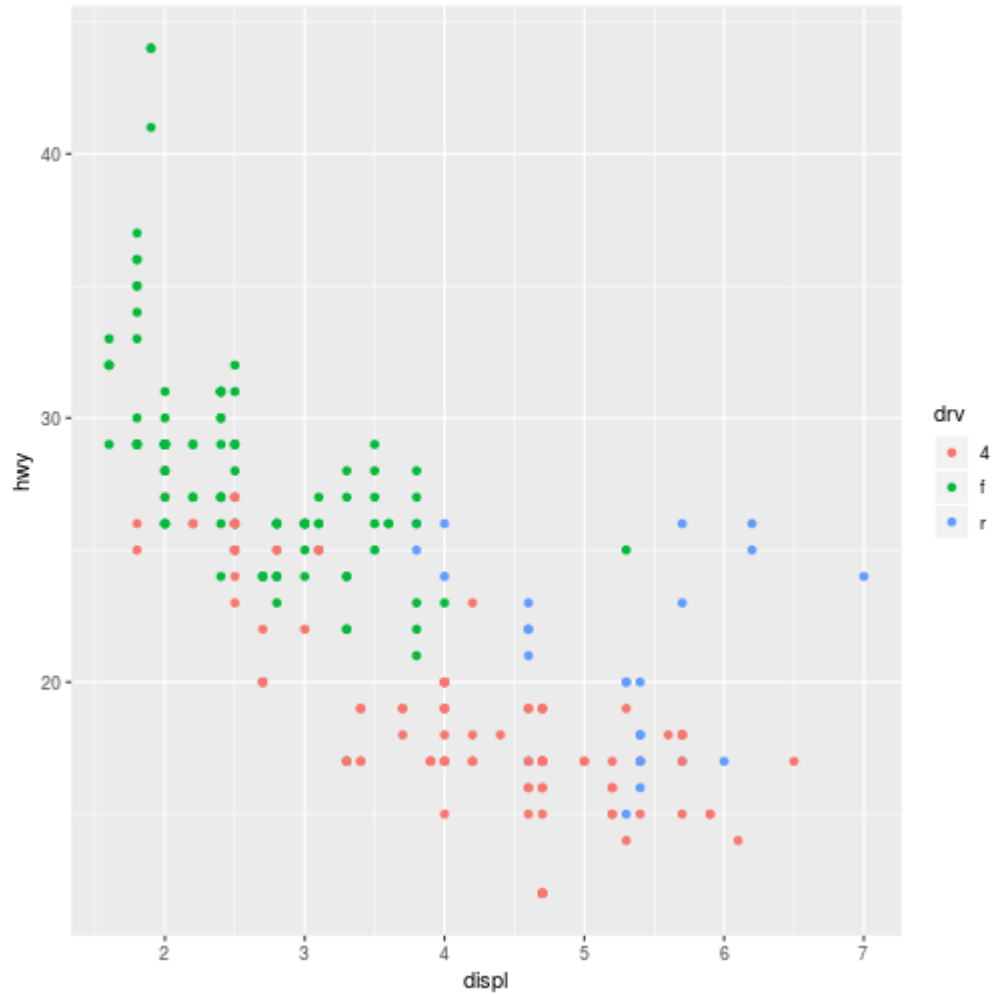
```
ggplot(data = <DATA>) +  
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

- Cambiando las secciones entre `<>` se pueden crear múltiples tipos de gráficos
- Añadiendo geoms con el operador `+` se pueden crear gráficos compuestos

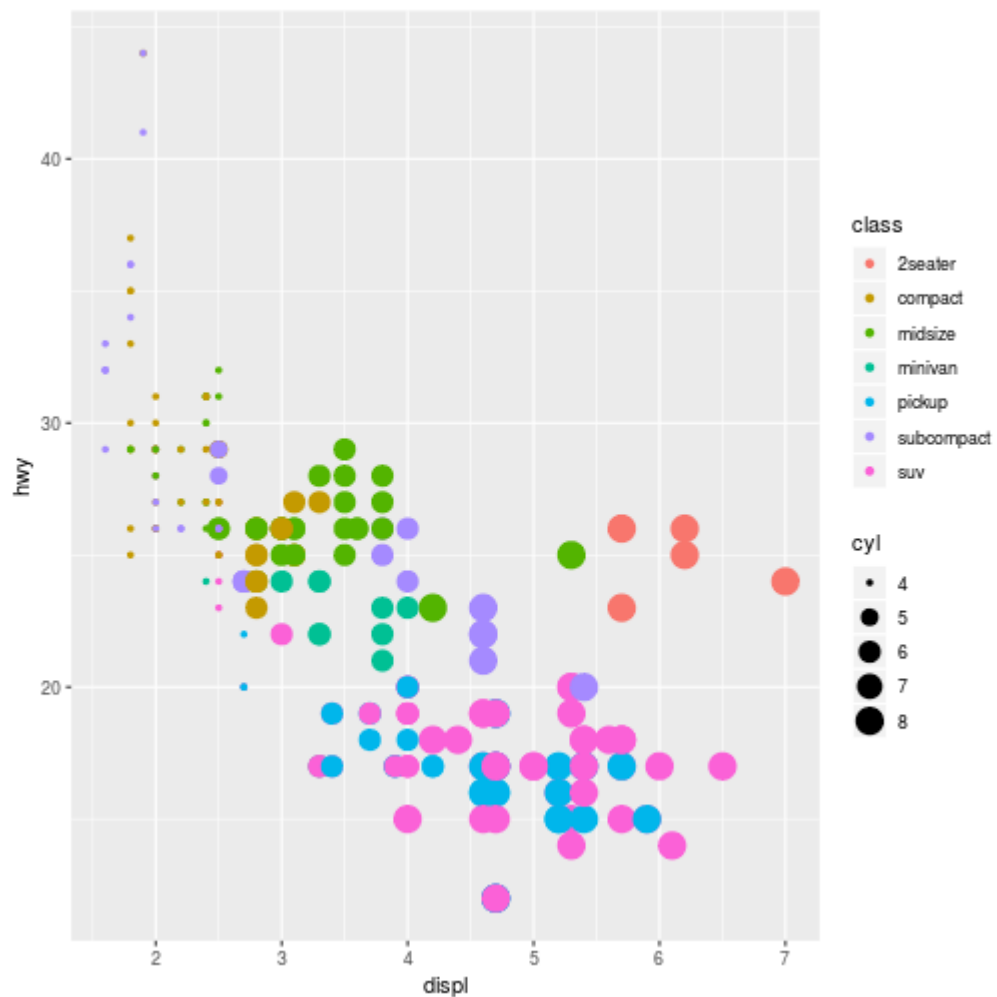
# Aesthetics

- El gráfico anterior representa dos variables, `displ` y `cyl`
- Variables adicionales se pueden asignar a distintas propiedades del gráfico (*aesthetics*)
- Algunos ejemplos son `color`, `shape`, `size`, `alpha`, etc.
- La escala y la leyenda se crean de forma automática

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



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

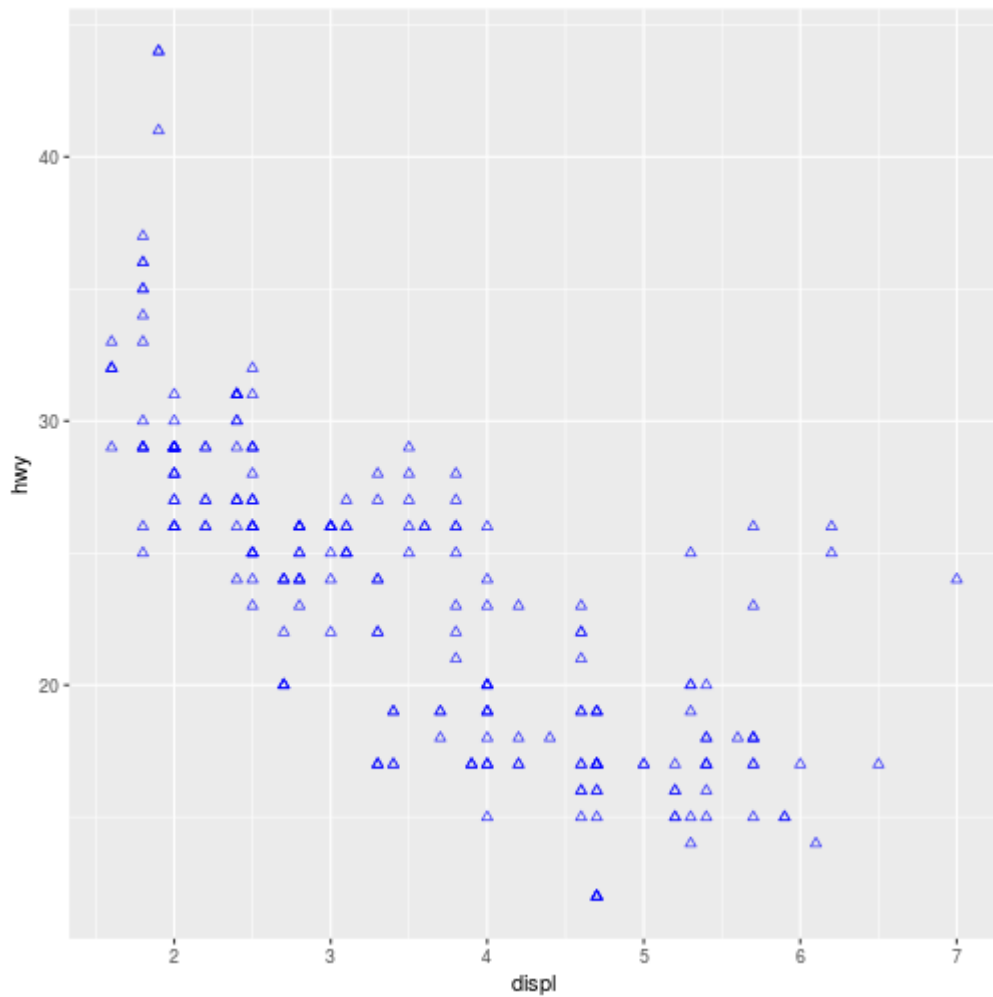




# Apariencia del gráfico

- Para cambiar la apariencia del gráfico, se les asigna un valor manualmente a las propiedades gráficas anteriores
- No transmiten información sobre una variable
- Tienen que estar **fuera** de la función `aes()`

```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy), color = "blue", alpha = 0.8, shape
```



# Facets

- Otra opción para representar variables adicionales son las facetas
- Cada faceta es un subgráfico realizado con un subconjunto de los datos

# facet\_wrap

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

# facet\_grid

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

# Geoms

- Objetos geométricos que se usan para representar la relación entre las variables `x` e `y`
- Algunos ejemplos son:
  - `geom_bar()` , barras
  - `geom_point()` , puntos
  - `geom_line()` , líneas
  - `geom_text()` , texto
  - ...
- Cada `geom` tiene una serie de propiedades gráficas que se pueden asignar a variables o modificar

# Múltiples geoms

Se pueden mostrar múltiples geoms añadiendo nuevas capas al gráfico

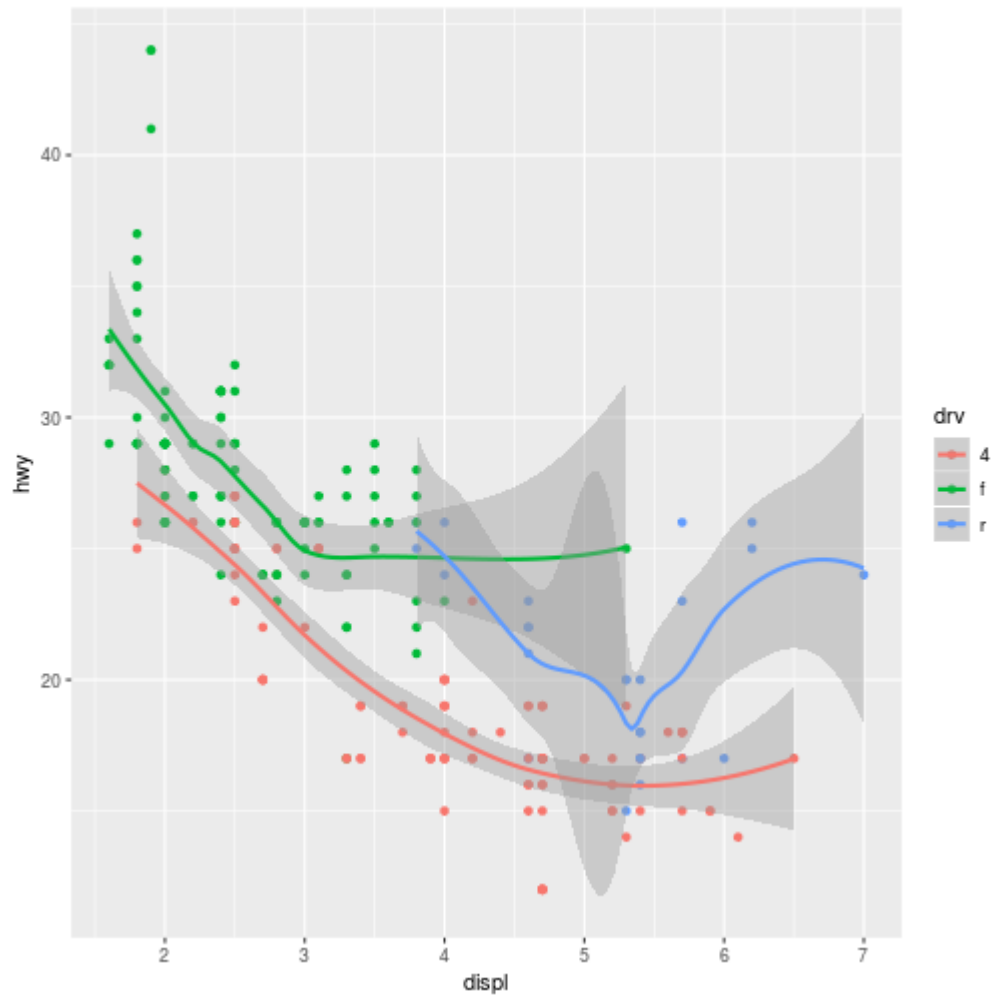
```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy)) +  
  geom_smooth(mapping = aes(x = displ, y = hwy))
```

# Ajustes globales

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



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



# Ajustes locales

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +  
  geom_point(mapping = aes(color = drv)) +  
  geom_smooth(linetype = 2)
```

# Transformaciones estadísticas

- Algunos `geom` calculan nuevas variables a representar a partir de las originales del data frame
- Un ejemplo es `geom_smooth()` , que ajusta un polinomio a los datos
- Para ver la transformación estadística de cada `geom` se puede consultar el valor por defecto del parámetro `stat` en la ayuda

# Ejemplo geom\_bar

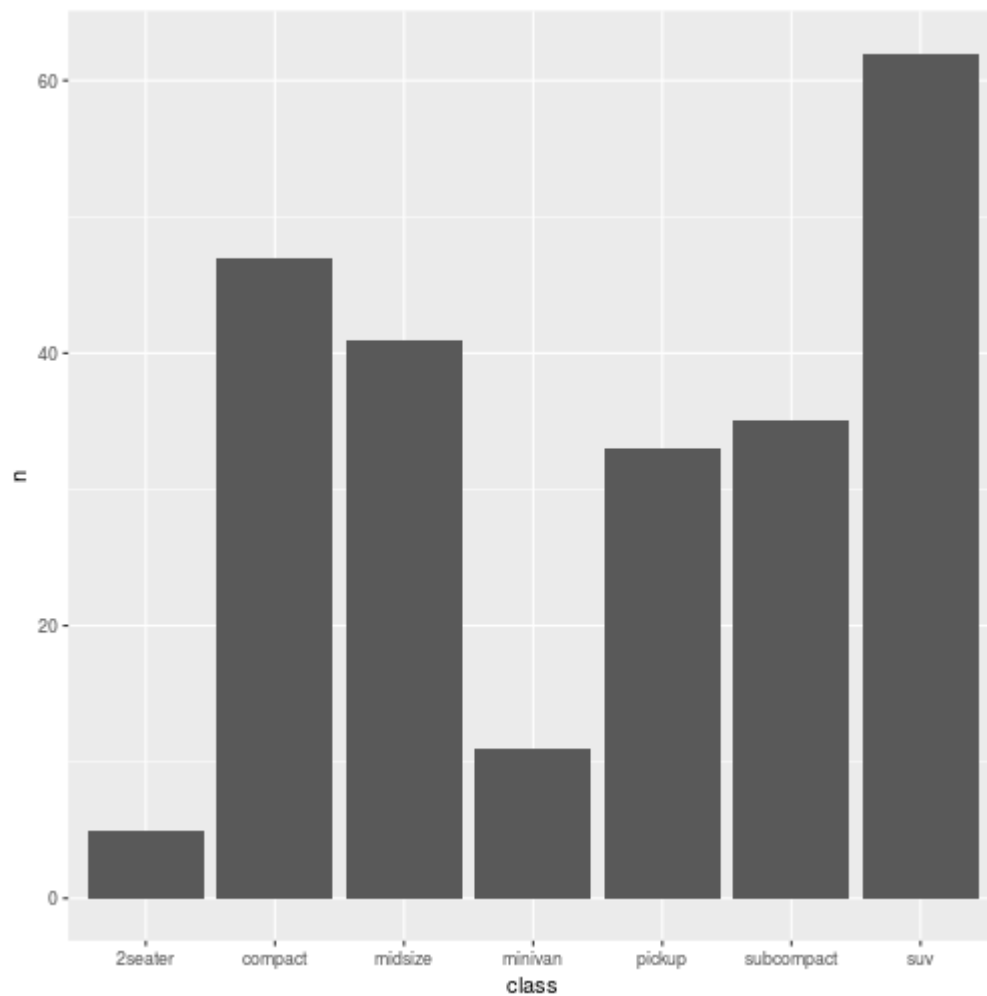
```
ggplot(data = mpg) +  
  geom_bar(aes(x = class))
```

# Cambiar stat por defecto

```
n_class <-  
  mpg %>%  
  group_by(class) %>%  
  summarize(n = n())  
  
n_class
```

```
## # A tibble: 7 x 2  
##   class      n  
##   <chr>    <int>  
## 1 2seater      5  
## 2 compact     47  
## 3 midsize     41  
## 4 minivan     11  
## 5 pickup      33  
## 6 subcompact  35  
## 7 suv         62
```

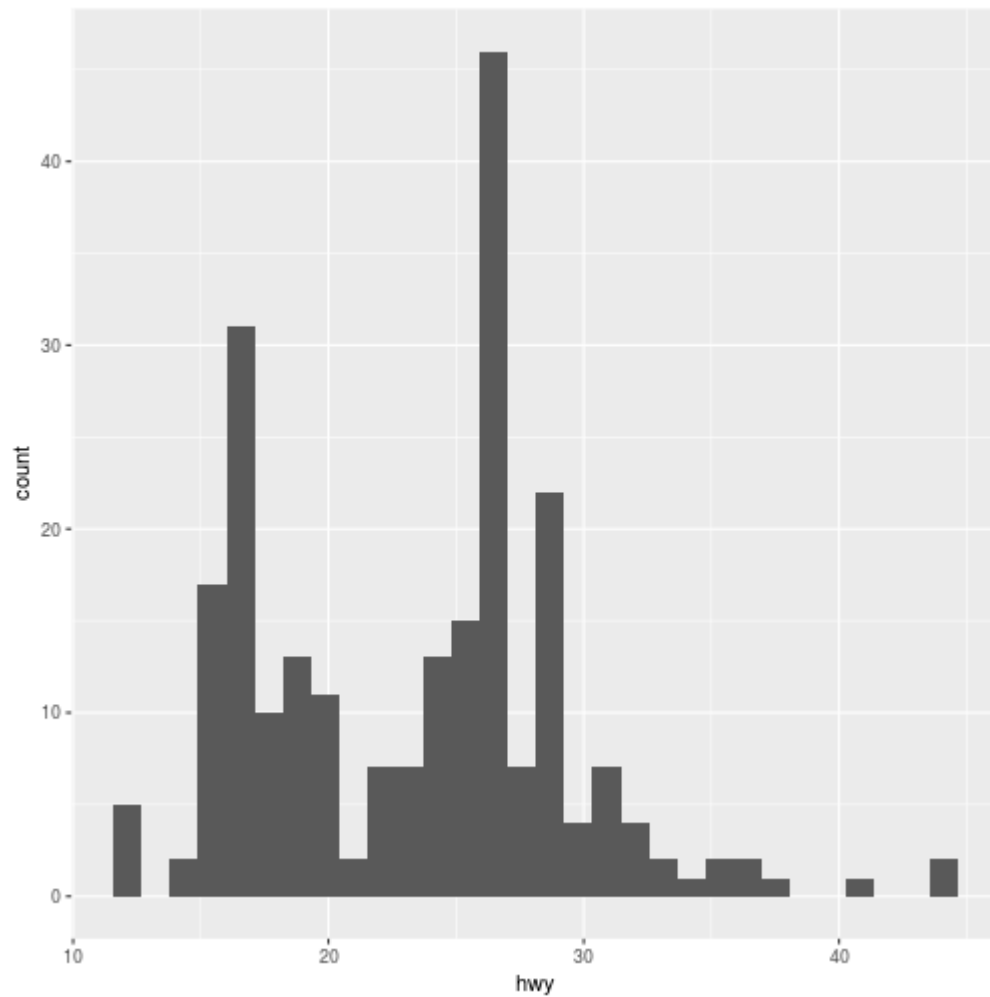
```
ggplot(data = n_class) +  
  geom_bar(aes(x = class, y = n), stat = "identity")
```



# Histograma

- Dada una variable continua:
  - Ordenar sus valores
  - Elegir número de intervalos
  - Contar cuantos valores hay en cada intervalo
  - Representar con barras
- La transformacion estadística se conoce como *binning*

```
ggplot(data = mpg) +  
  geom_bar(mapping = aes(x = hwy), stat = "bin")
```





# Resultado transformación

Las variables resultado de la transformación son accesibles como `....`

```
ggplot(data = mpg) +  
  geom_bar(mapping = aes(x = hwy, y = ..density..), stat = "bin")
```

Juntando lo anterior podríamos, por ejemplo, representar un histograma con puntos en vez de barras

```
ggplot(data = mpg) +  
  geom_line(mapping = aes(x = hwy, y = ..count..), stat = "bin")
```

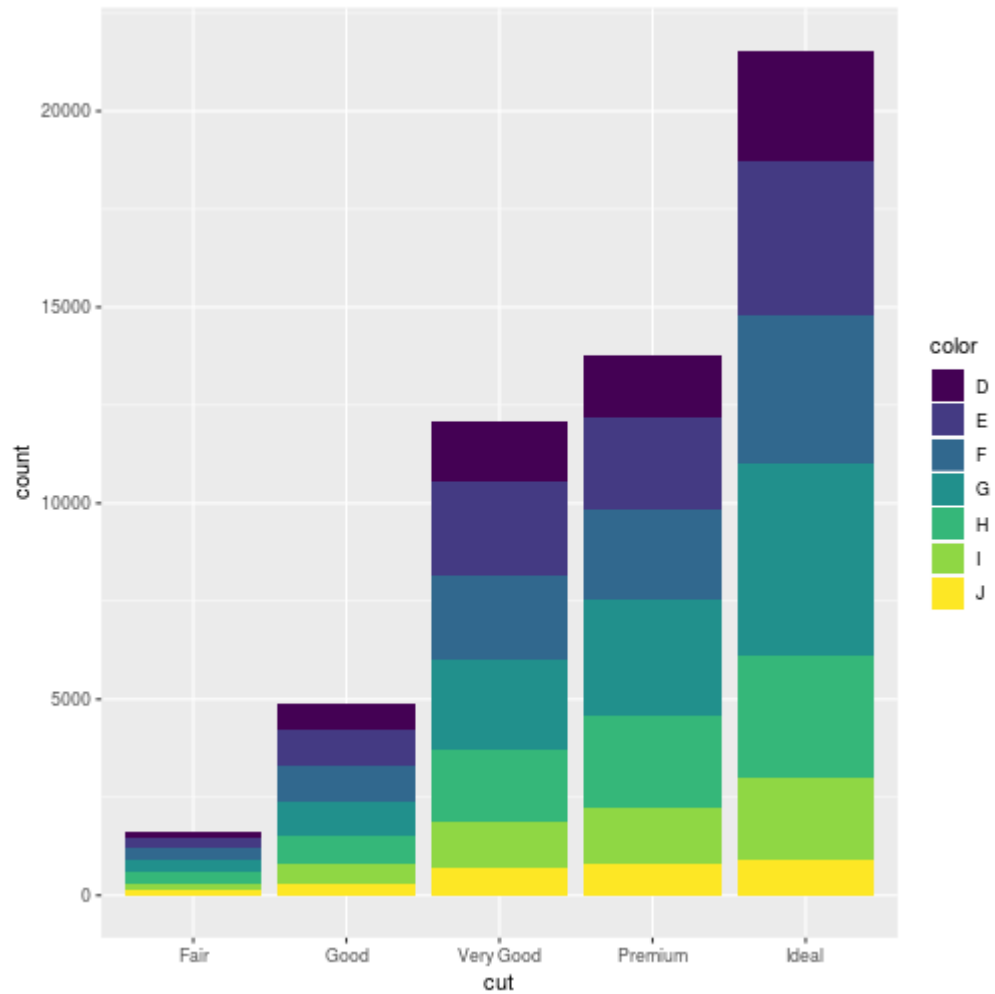
# geom\_hist

```
ggplot(data = mpg) +  
  geom_histogram(mapping = aes(x = hwy))
```

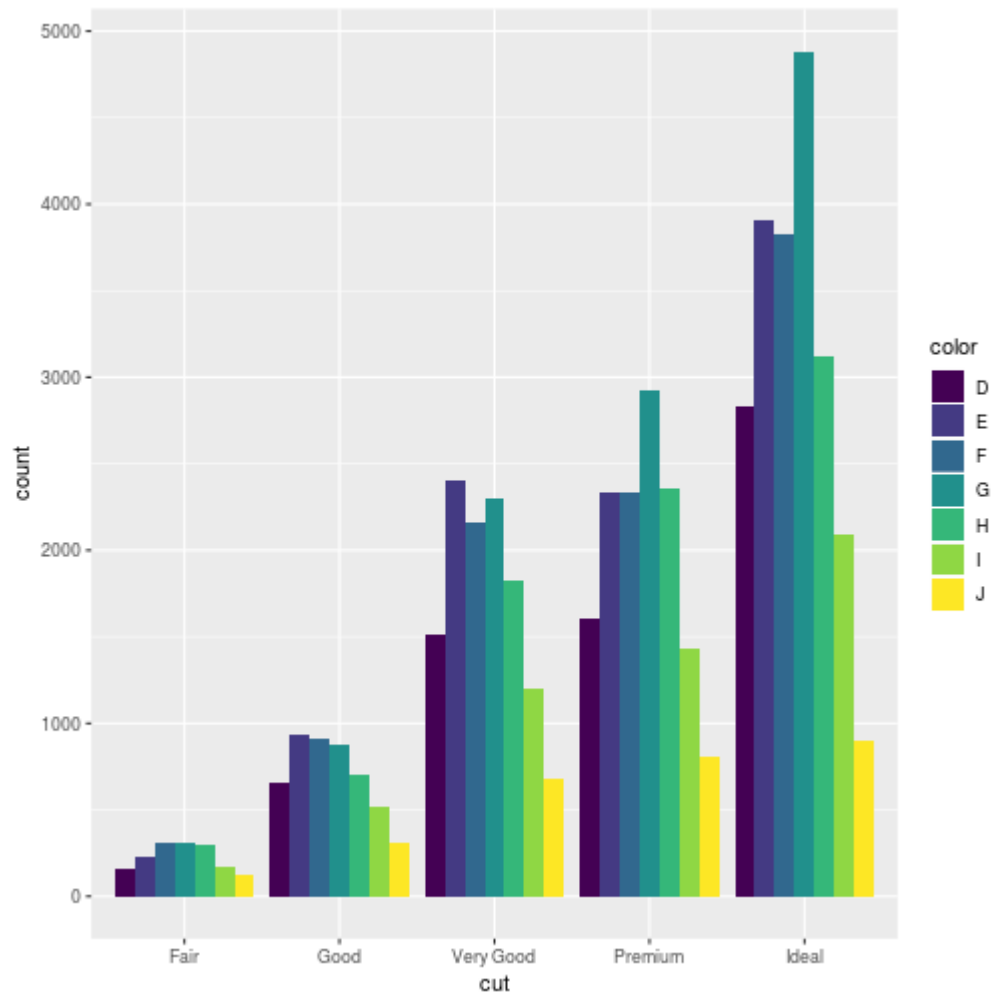
# Ajustes de posición

- Ciertos `geom`s tienen un ajuste opcional de posición
- En `geom_bar()` su valor por defecto es `stack`
- Otros ajustes posibles son `dodge` y `fill`
- En `geom_point()` su valor por defecto es `identity`
- Otro valor posible es `jitter`

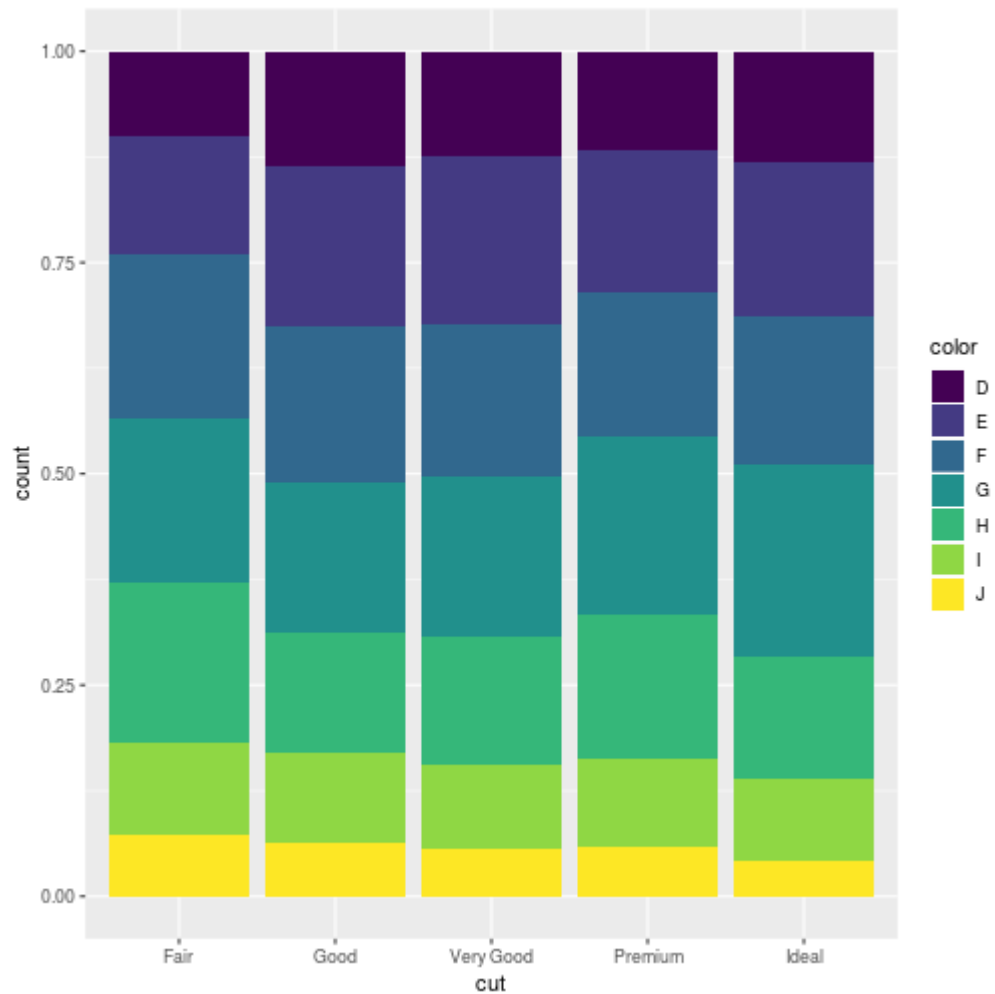
```
ggplot(data = diamonds) +  
  geom_bar(mapping = aes(x = cut, fill = color))
```



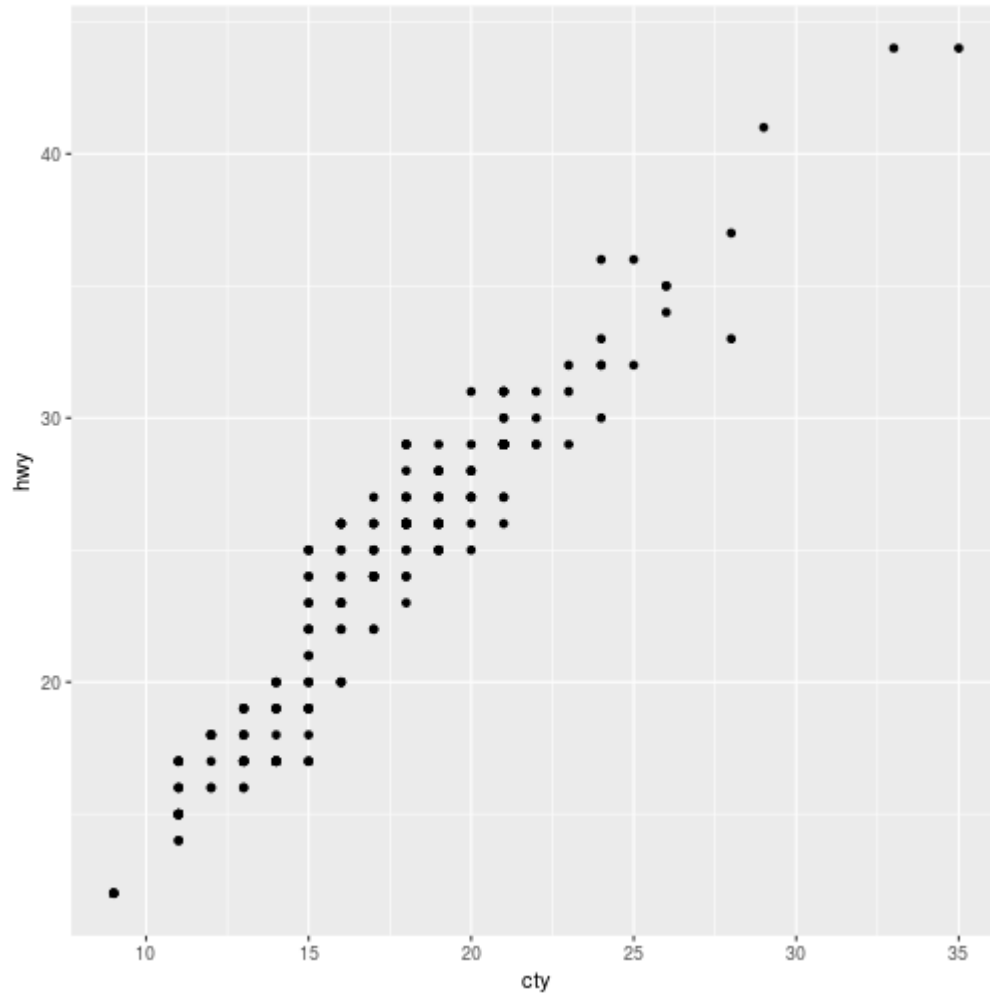
```
ggplot(data = diamonds) +  
  geom_bar(mapping = aes(x = cut, fill = color), position = "dodge")
```



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



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





```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = cty, y = hwy), position = "jitter")
```

