

ggplot2

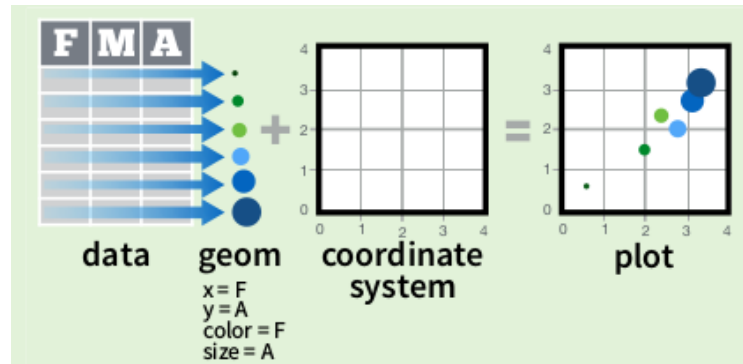
Fundamentos lenguajes: R

Alberto Torres e Irene Rodríguez

2019-12-14

Introducción

- Implementa una **gramática de gráficos** en R.
- Divide un gráfico en sus componentes esenciales:
 - Un conjunto de datos.
 - Un conjunto de marcas visuales (puntos, líneas, barras) y propiedades asociadas a ellas (color, tamaño, tipo, etc.).
 - Un sistema de coordenadas.



- Múltiples ventajas con respecto a los gráficos de R base
 - Leyenda automática
 - Facetas
 - ...

Introducción (cont.)

- Los capítulos de *data visualization* y *graphics for communication* del libro *R for data science* son una buena forma de empzar.
- La *cheatsheet de ggplot2* es muy útil como resumen, así como la *documentación de referencia de ggplot2* que contiene muchos ejemplos.

Gramática de gráficos

- Para generar un gráfico en ggplot, se usan las funciones `qplot()` o `ggplot()`.
 - `qplot()` es una versión simplificada de `ggplot()` que permite definir los datos a representar, coordenadas y la forma en qué se representar (líneas, puntos...), y provee muchos valores por defecto.
- Por su flexibilidad, veremos `ggplot()`.

Gramática de gráficos (cont.)

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

Begins a plot that you finish by adding layers to. No defaults, but provides more control than `qplot()`.

data

```
ggplot(mpg, aes(hwy, cty)) +  
  geom_point(aes(color = cyl)) +  
  geom_smooth(method = "lm") +  
  coord_cartesian() +  
  scale_color_gradient() +  
  theme_bw()
```

add layers,
elements with +

layer = geom +
default stat +
layer specific
mappings

additional
elements

Add a new layer to a plot with a **geom_*()** or **stat_*()** function. Each provides a geom, a set of aesthetic mappings, and a default stat and position adjustment.

- *data* data frame
- *mapping* se define con `aes()` (*aesthetics*) y describe como las variables de un data frame se asignan a propiedades visuales
- *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

Plantilla

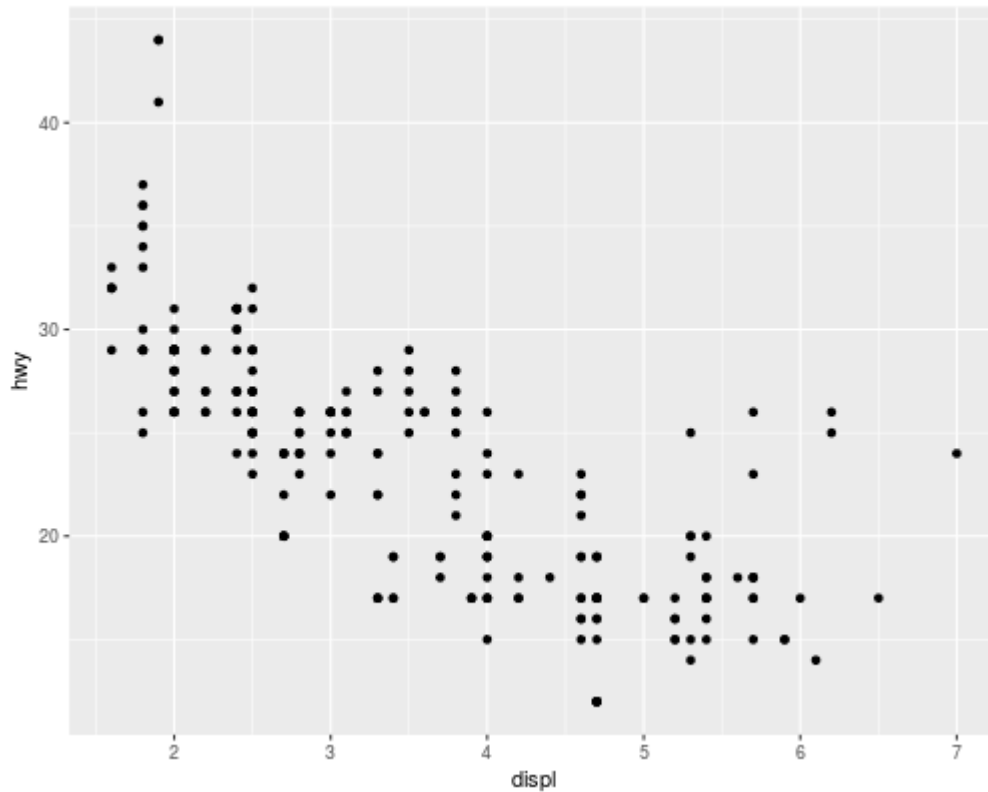
- El gráfico más sencillo consta como mínimo de los siguientes componentes [**Fuente**]:

```
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

Ejemplo

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

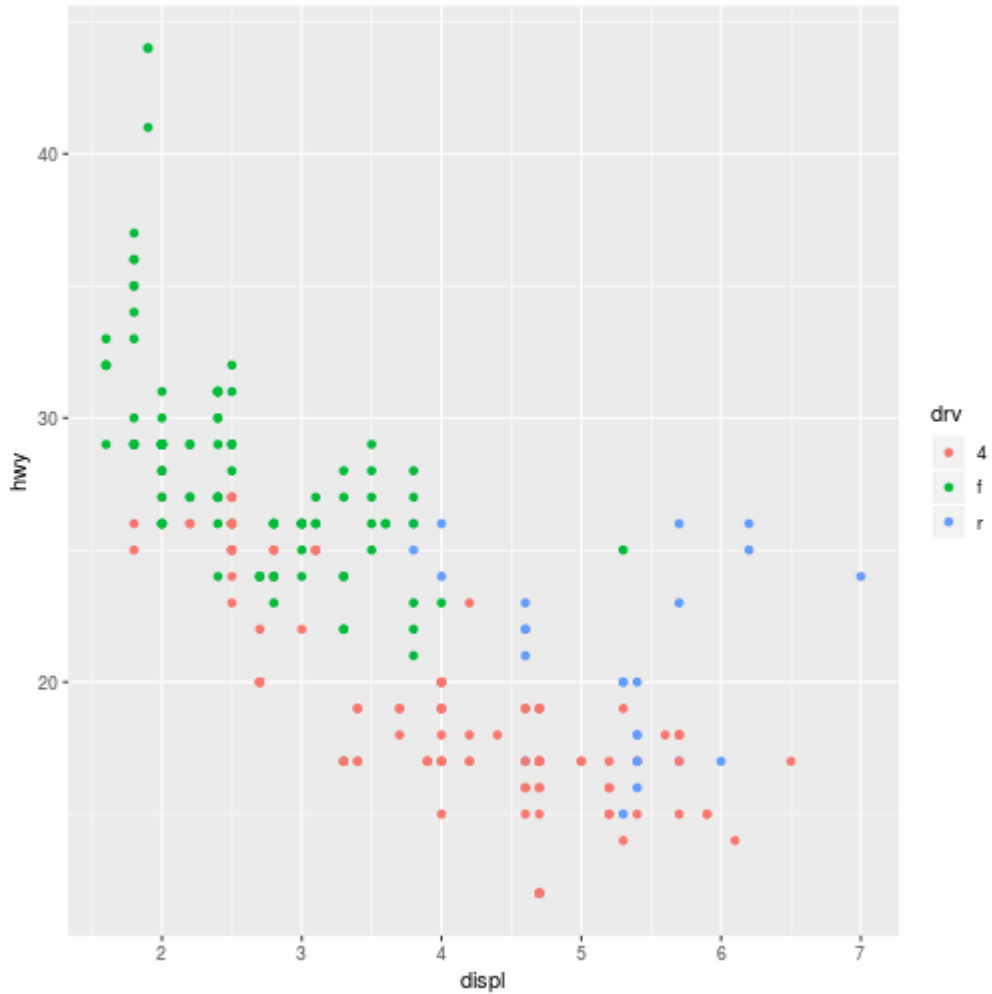


Aesthetics

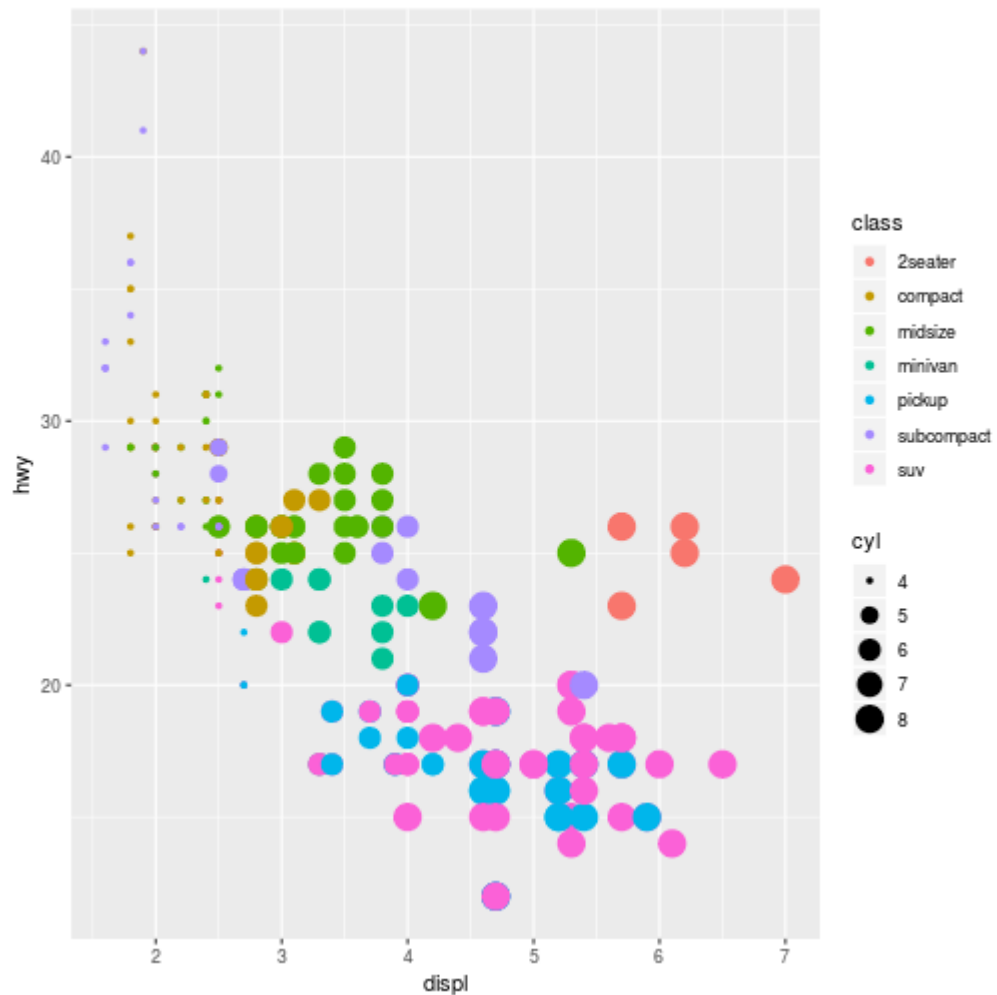
```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy))
```

- 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))
```



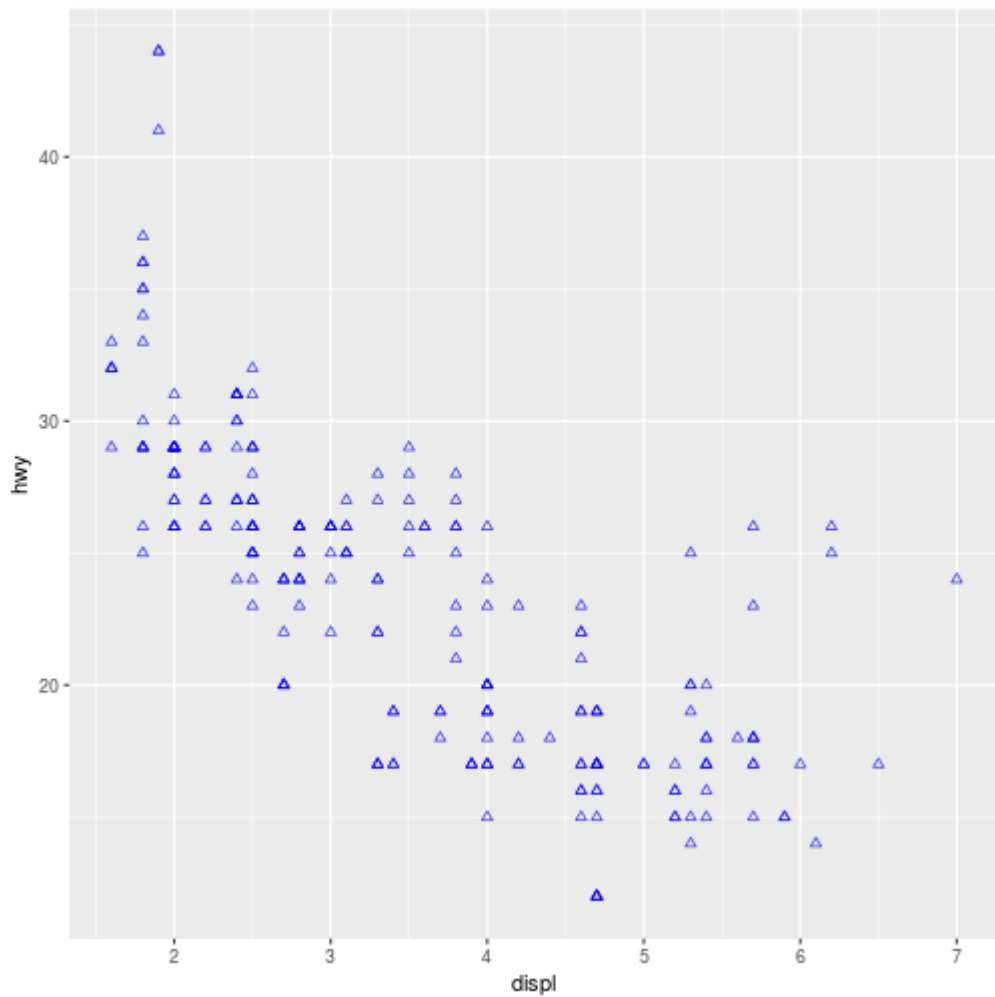
```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy, color = class, size = cyl))
```



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
```



Geoms

- Objetos geométricos que se usan para representar la relación entre las variables `x` e `y`
- Algunos ejemplos son:
 - `geom_bar()` , barras
 - `geom_histogram()` , histograma
 - `geom_density()` , función de densidad
 - `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

Geoms (cont.)

One Variable

Continuous

```
a <- ggplot(mpg, aes(hwy))
```



```
a + geom_area(stat = "bin")
```

x, y, alpha, color, fill, linetype, size
b + geom_area(aes(y = ..density..), stat = "bin")



```
a + geom_density(kernel = "gaussian")
```

x, y, alpha, color, fill, linetype, size, weight
b + geom_density(aes(y = ..county..))



```
a + geom_dotplot()
```

x, y, alpha, color, fill



```
a + geom_freqpoly()
```

x, y, alpha, color, linetype, size
b + geom_freqpoly(aes(y = ..density..))



```
a + geom_histogram(binwidth = 5)
```

x, y, alpha, color, fill, linetype, size, weight
b + geom_histogram(aes(y = ..density..))

Discrete

```
b <- ggplot(mpg, aes(fl))
```



```
b + geom_bar()
```

x, alpha, color, fill, linetype, size, weight

Two Variables

Continuous X, Continuous Y

```
f <- ggplot(mpg, aes(cty, hwy))
```



```
f + geom_blank()
```



```
f + geom_jitter()
```

x, y, alpha, color, fill, shape, size



```
f + geom_point()
```

x, y, alpha, color, fill, shape, size



```
f + geom_quantile()
```

x, y, alpha, color, linetype, size, weight



```
f + geom_rug(sides = "bl")
```

alpha, color, linetype, size



```
f + geom_smooth(model = lm)
```

x, y, alpha, color, fill, linetype, size, weight



```
f + geom_text(aes(label = cty))
```

x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

Discrete X, Continuous Y

```
g <- ggplot(mpg, aes(class, hwy))
```



```
g + geom_bar(stat = "identity")
```

x, y, alpha, color, fill, linetype, size, weight



```
g + geom_boxplot()
```

lower, middle, upper, x, ymax, ymin, alpha, color, fill, linetype, shape, size, weight



```
g + geom_dotplot(binaxis = "y", stackdir = "center")
```

x, y, alpha, color, fill



```
g + geom_violin(scale = "area")
```

x, y, alpha, color, fill, linetype, size, weight

Discrete X, Discrete Y

```
h <- ggplot(diamonds, aes(cut, color))
```



```
h + geom_jitter()
```

x, y, alpha, color, fill, shape, size

Continuous Bivariate Distribution

```
i <- ggplot(movies, aes(year, rating))
```



```
i + geom_bin2d(binwidth = c(5, 0.5))
```

xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size, weight



```
i + geom_density2d()
```

x, y, alpha, colour, linetype, size



```
i + geom_hex()
```

x, y, alpha, colour, fill, size

Continuous Function

```
j <- ggplot(economics, aes(date, unemploy))
```



```
j + geom_area()
```

x, y, alpha, color, fill, linetype, size



```
j + geom_line()
```

x, y, alpha, color, linetype, size



```
j + geom_step(direction = "hv")
```

x, y, alpha, color, linetype, size

Visualizing error

```
df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
k <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))
```



```
k + geom_crossbar(fatten = 2)
```

x, y, ymax, ymin, alpha, color, fill, linetype, size



```
k + geom_errorbar()
```

x, ymax, ymin, alpha, color, linetype, size, width (also `geom_errorbarh()`)



```
k + geom_linerange()
```

x, ymin, ymax, alpha, color, linetype, size



```
k + geom_pointrange()
```

x, y, ymin, ymax, alpha, color, fill, linetype, shape, size

Maps

```
data <- data.frame(murder = USArrests$Murder, state = tolower(rownames(USArrests)))
map <- map_data("state")
l <- ggplot(data, aes(fill = murder))
```



```
l + geom_map(aes(map_id = state), map = map) + expand_limits(x = map$long, y = map$lat)
```

map_id, alpha, color, fill, linetype, size

Múltiples geoms

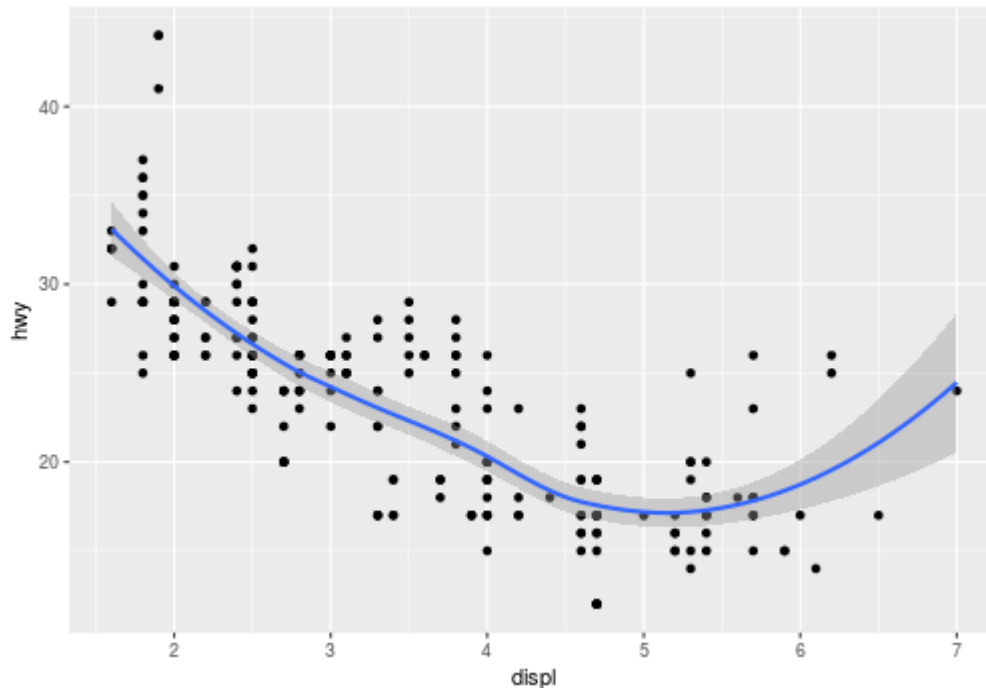
Se pueden mostrar múltiples geoms añadiendo nuevas capas al gráfico:

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

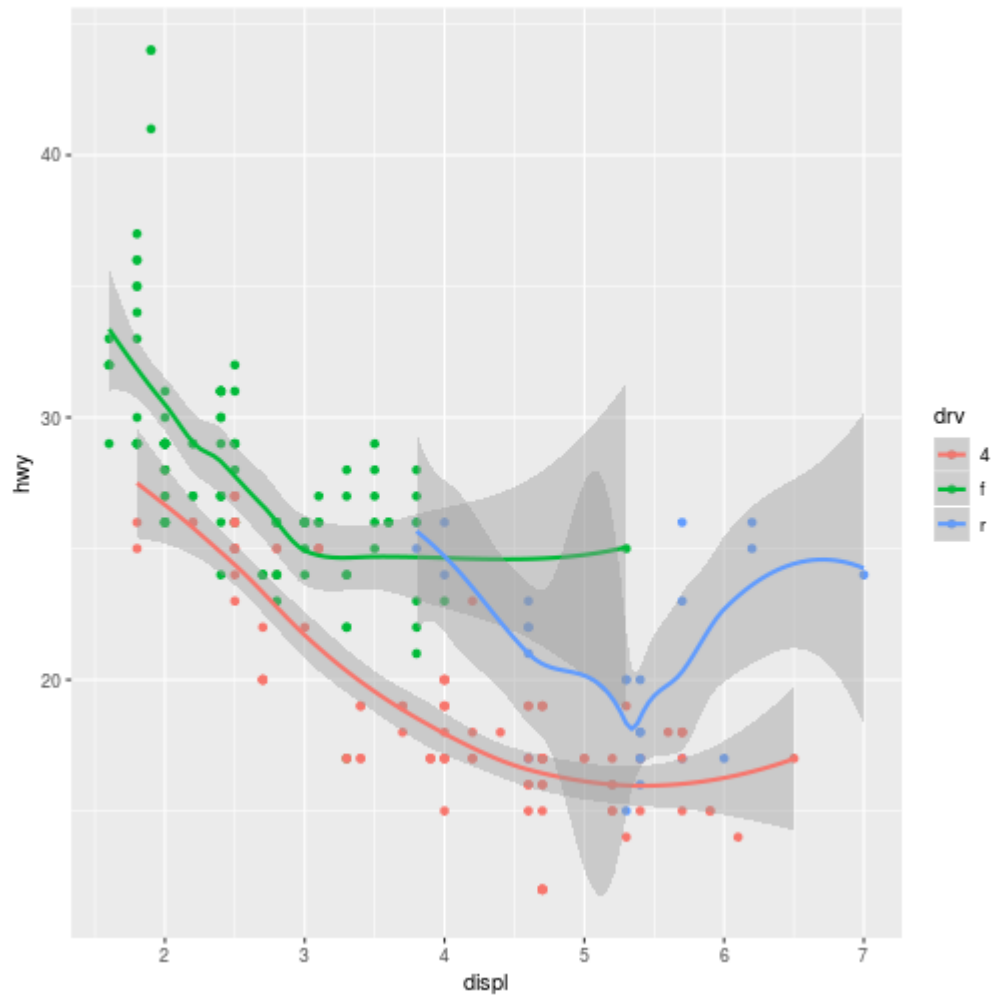
Ajustes globales

- Se puede configurar el `aes` en la llamada a `ggplot()` y las funciones tipo `geom` tomarán ese `aes` en caso de que no se sobrescriban los atributos del `aes` en la llamada a la función `geom`.

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

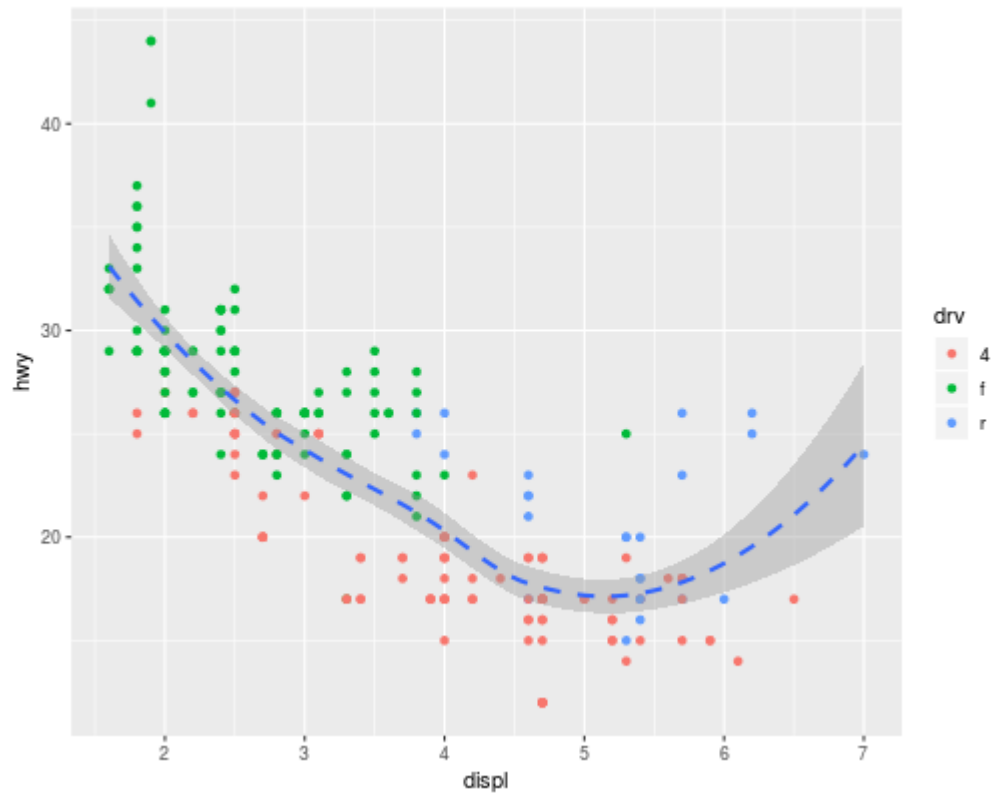



```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = drv)) +  
  geom_point() +  
  geom_smooth()
```



Ajustes locales

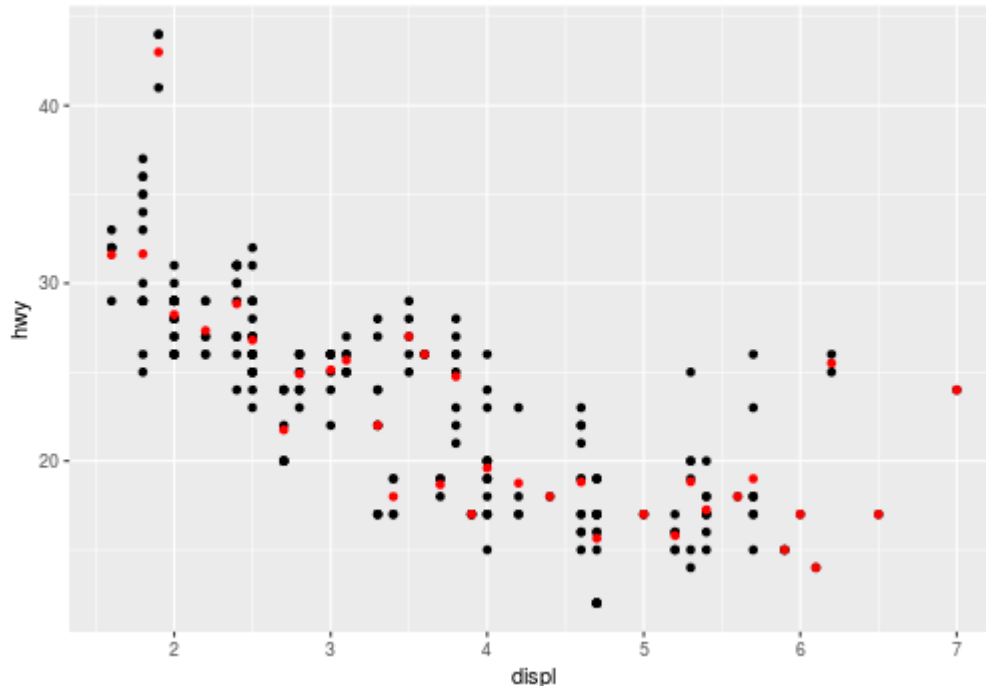
```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +  
  geom_point(mapping = aes(color = drv)) +  
  geom_smooth(linetype = 2)
```



Ajustes locales (cont.)

- También es posible cambiar los datos a representar en un `geom`.

```
mymean <- mpg %>% group_by(displ) %>% summarize(media=mean(hwy))  
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +  
  geom_point() +  
  geom_point(data=mymean, mapping=aes(x=displ,y=media),color="red")
```

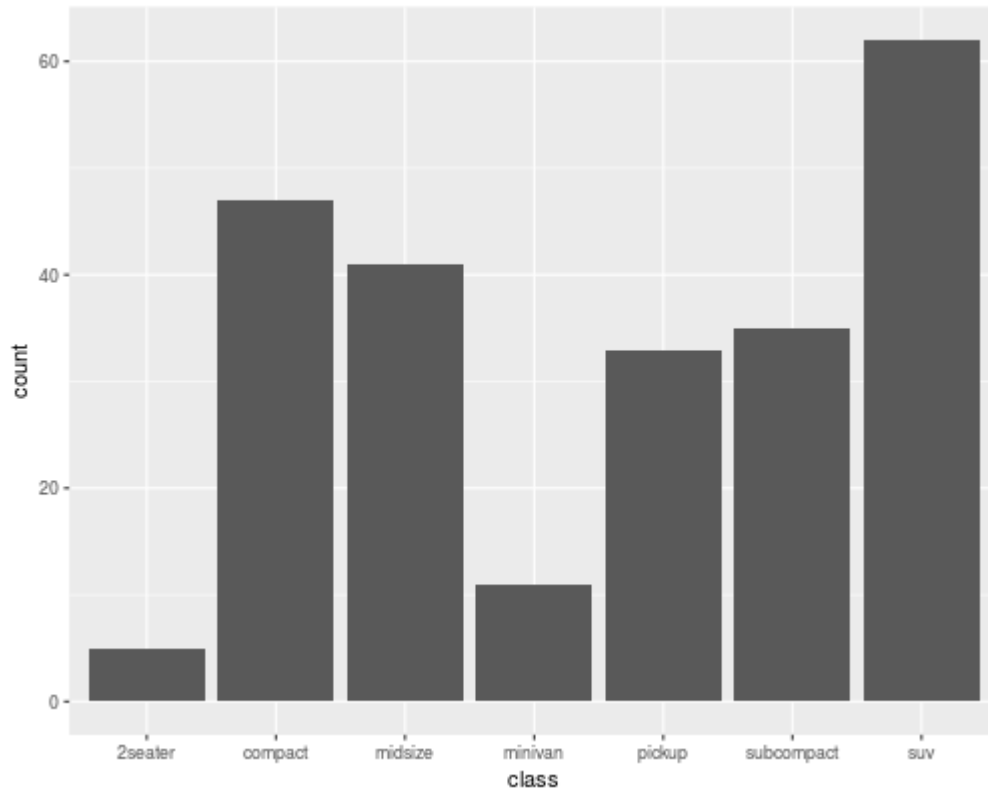


Transformaciones estadísticas

- 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))
```

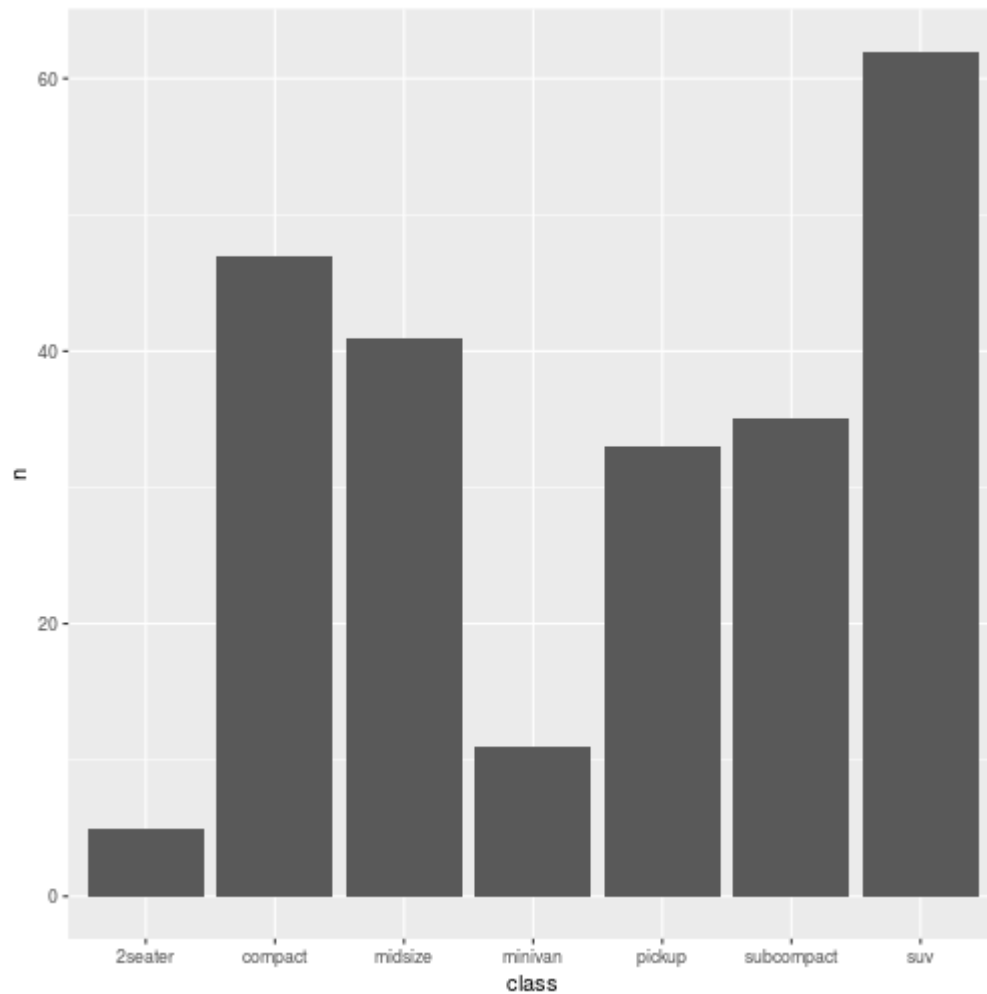


Cambiar stat por defecto

```
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
```

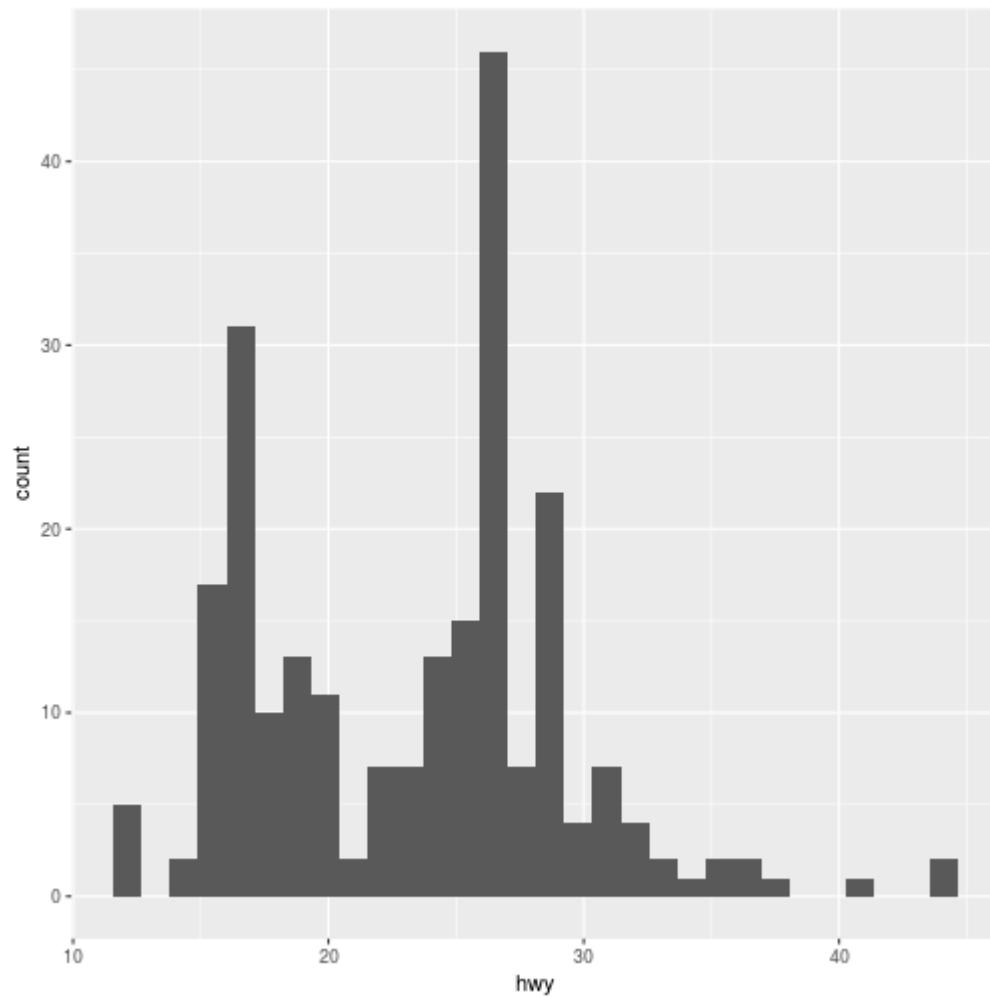
```
ggplot(data = n_class) +  
  geom_bar(aes(x = class, y = n), stat = "identity")
```



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")
```

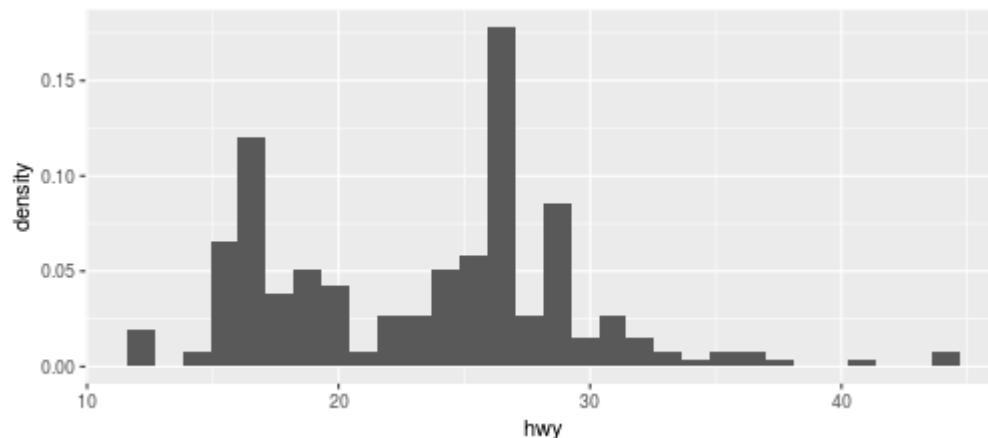


Resultado transformación

Las variables resultado de la transformación son accesibles como `..<NOMBRE>..`

- `..count..` : número de puntos en cada bin
- `..ncount..` : número de puntos en cada bin normalizados por el máximo de conteos sobre todos los bins.
- `..density..` : función de densidad
- `..ndensity..` : función de densidad normalizada por el máximo valor de densidad sobre todos los bins.

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

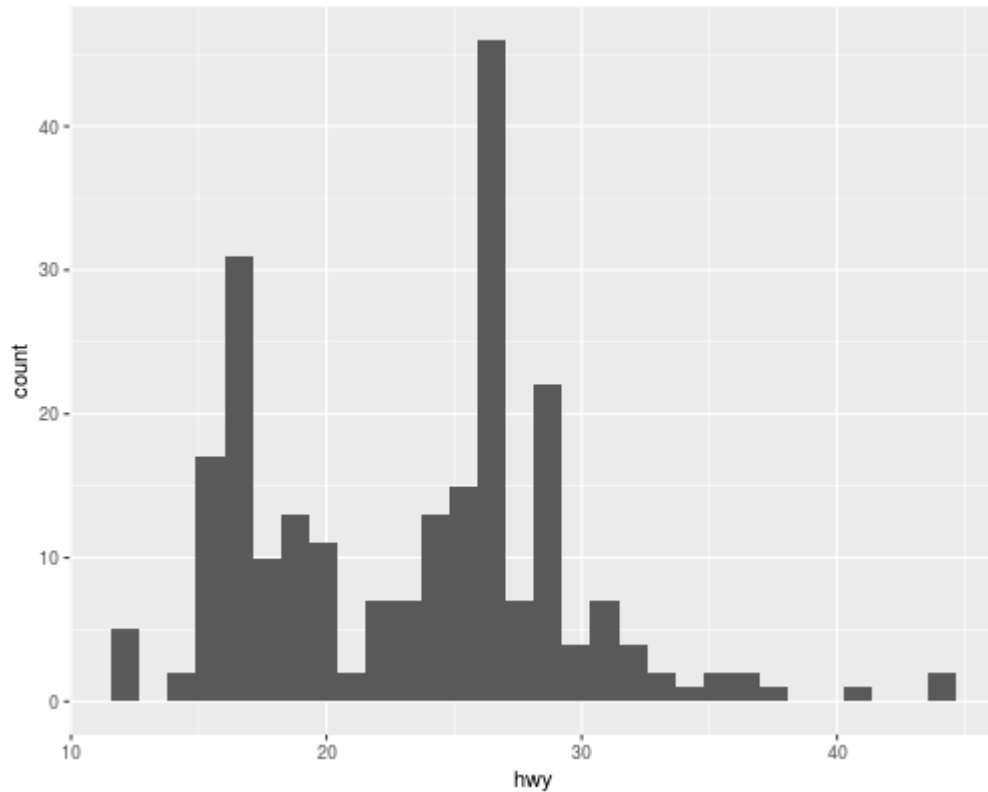


Juntando lo anterior podríamos, por ejemplo, representar un histograma con puntos en vez de barras

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

geom_histogram

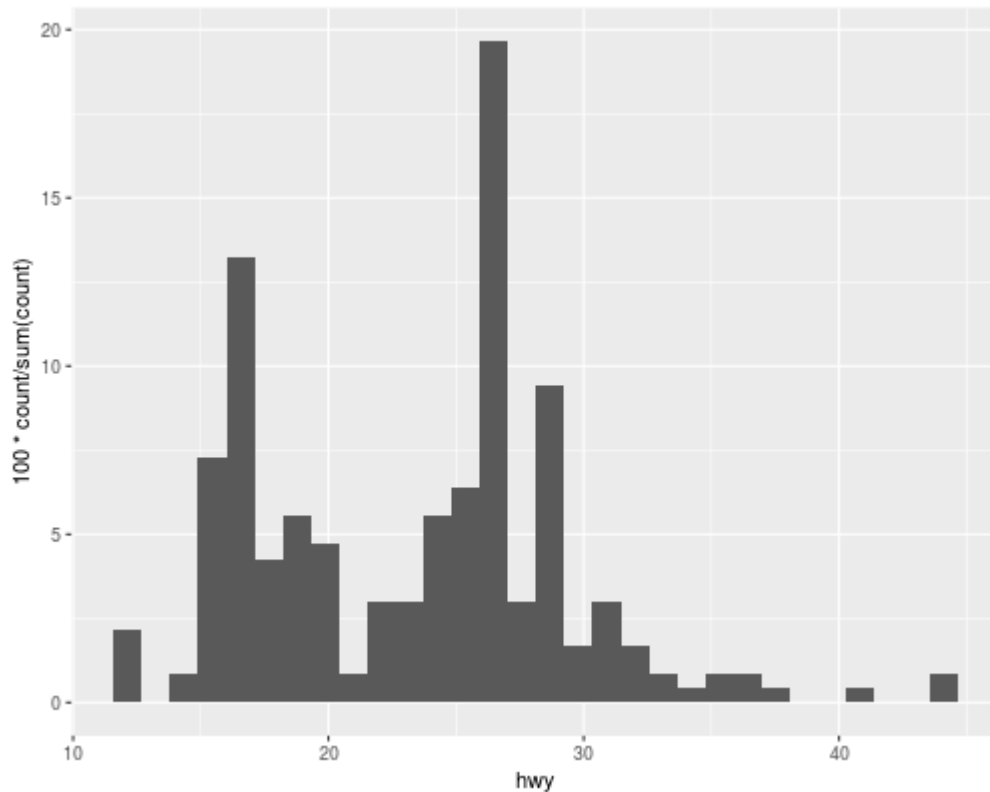
```
ggplot(data = mpg) +  
  geom_histogram(mapping = aes(x = hwy))
```



geom_histogram (cont.)

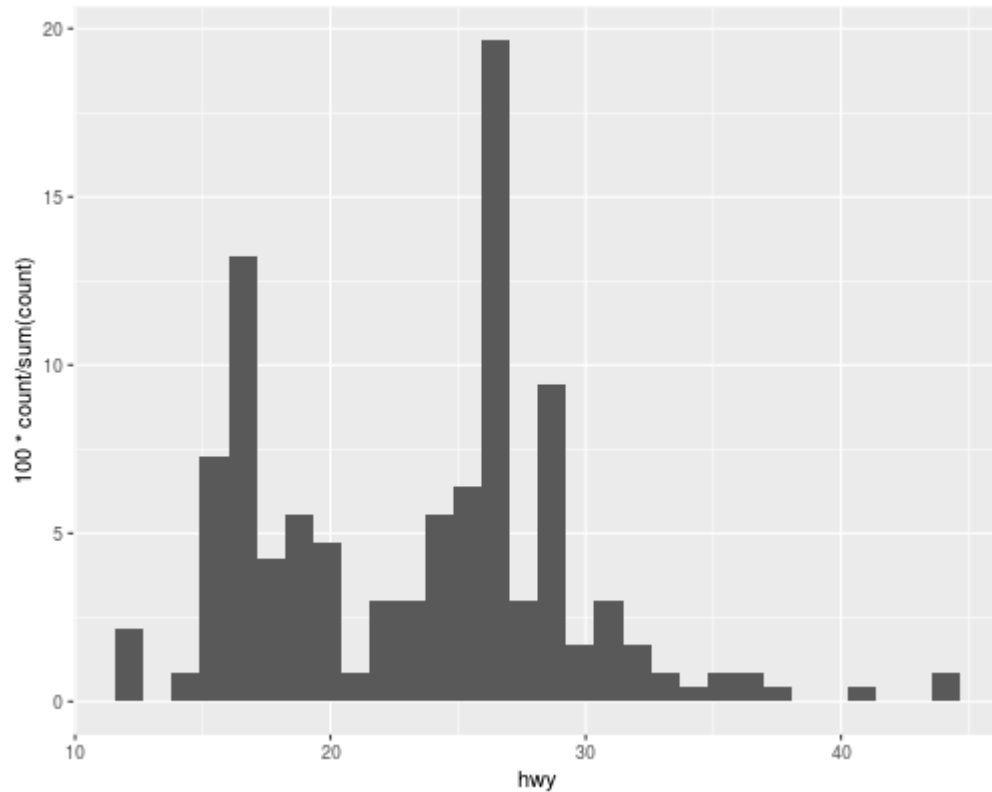
- Generamos el histograma anterior normalizado y en porcentaje.

```
ggplot(data = mpg) +  
  geom_histogram(mapping = aes(x = hwy, y=100*..count../sum(..count..)))
```



- De forma equivalente:

```
ggplot(data = mpg) +  
  geom_bar(mapping = aes(x = hwy, y=100*..count../sum(..count..)), stat="bin")
```



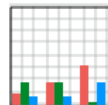
Ajustes de posición

- Ciertos `geom`s tienen un ajuste opcional de posición.
 - En `geom_bar()` su valor por defecto es `stack`.
 - En `geom_point()` su valor por defecto es `identity`.

Position Adjustments

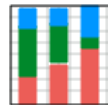
Position adjustments determine how to arrange geoms that would otherwise occupy the same space.

```
s <- ggplot(mpg, aes(fl, fill = drv))
```



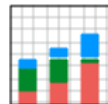
s + geom_bar(position = "dodge")

Arrange elements side by side



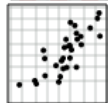
s + geom_bar(position = "fill")

Stack elements on top of one another, normalize height



s + geom_bar(position = "stack")

Stack elements on top of one another



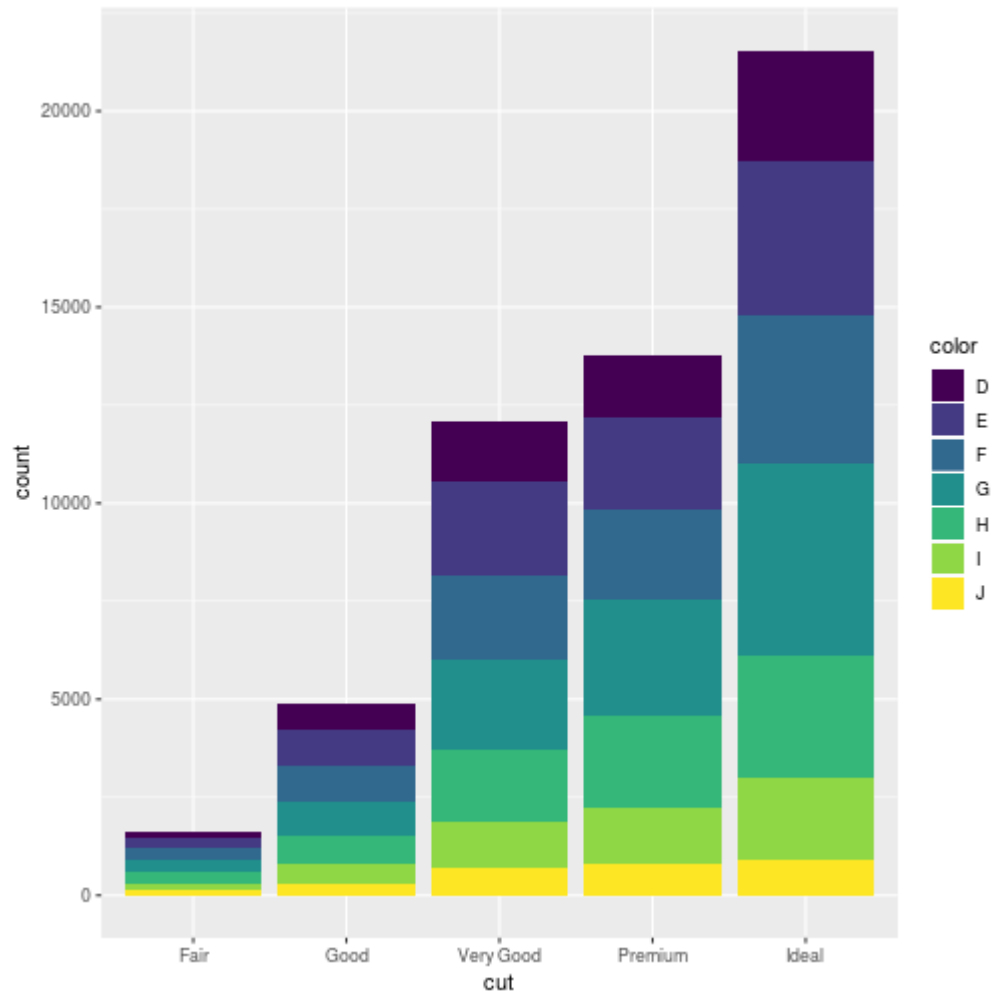
f + geom_point(position = "jitter")

Add random noise to X and Y position of each element to avoid overplotting

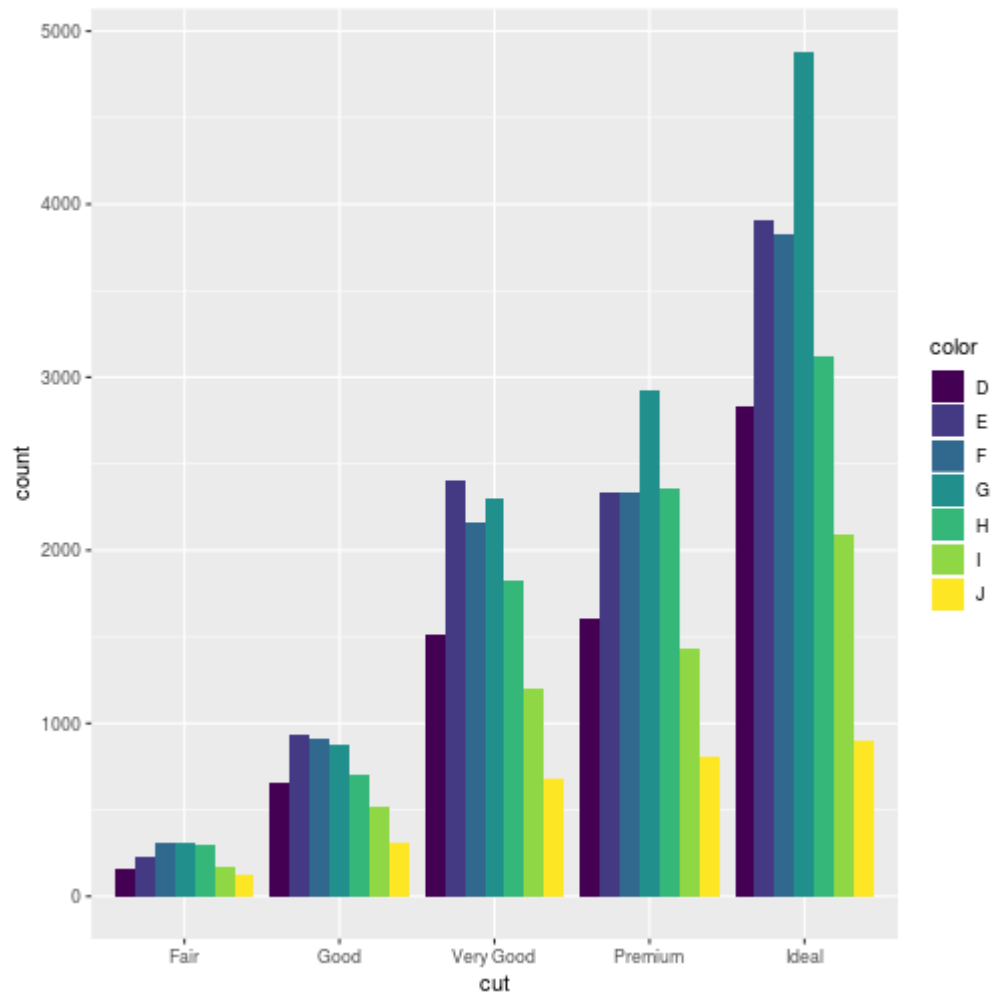
Each position adjustment can be recast as a function with manual **width** and **height** arguments

```
s + geom_bar(position = position_dodge(width = 1))
```

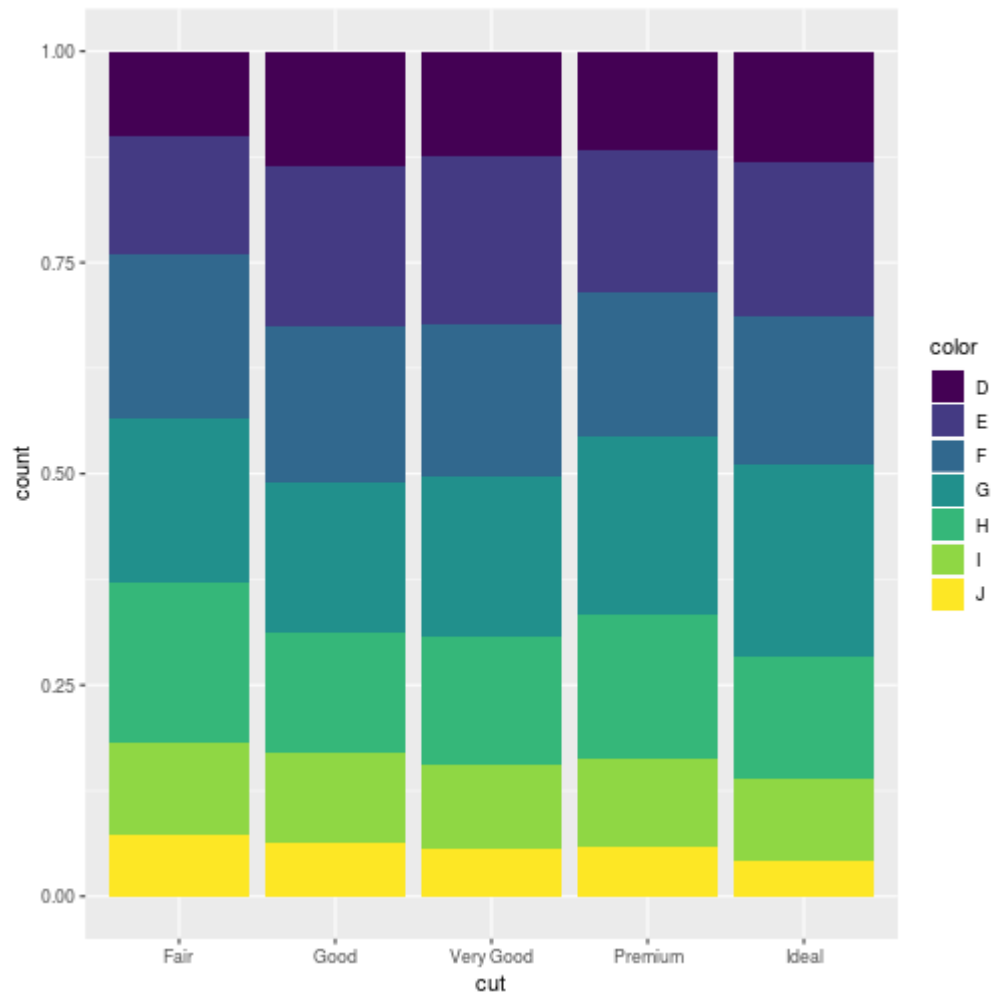
```
ggplot(data = diamonds) +  
  geom_bar(mapping = aes(x = cut, fill = color))
```



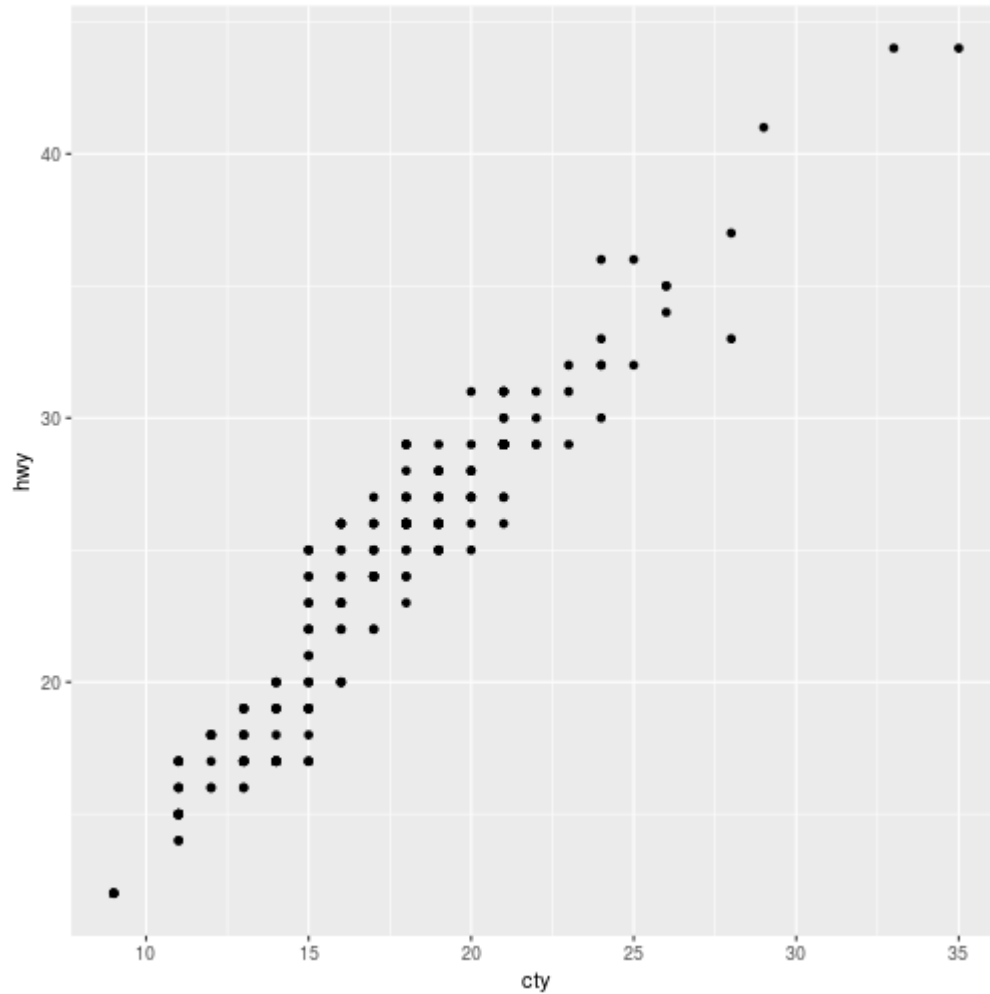

```
ggplot(data = diamonds) +  
  geom_bar(mapping = aes(x = cut, fill = color), position = "dodge")
```



```
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")
```

