ggplot

Rodrigo Negrete Pérez

September 3, 2022

8. Gráficos de barras
9. Histograma y Densidad
10. ggsave()

Section 1

Intro a ggplot

Parte del tidyverse

- ggplot es parte del tidyverse
- Si bien R tiene gráficas base:
  - no son tan presonalizazbles
  - ggplot puede vincular estética a variable
- Ampliamente utilizado por su sintaxis

mtcars

- Usemos la tradicional base de datos de mtcars

```
cars<- mpg
```

	mpg	cyl	disp	hp	drat	wt	qsec
Mazda RX4	21.00	6.00	160.00	110.00	3.90	2.62	16.46
Mazda RX4 Wag	21.00	6.00	160.00	110.00	3.90	2.88	17.02
Datsun 710	22.80	4.00	108.00	93.00	3.85	2.32	18.61
Hornet 4 Drive	21.40	6.00	258.00	110.00	3.08	3.21	19.44
Hornet Sportabout	18.70	8.00	360.00	175.00	3.15	3.44	17.02
Valiant	18.10	6.00	225.00	105.00	2.76	3.46	20.22

Section 2

Sintaxis

Sintaxis

```
ggplot(data=<DATA>)+
  geom_<<funcion>>(mapping= aes(x= <<x>> , y= <<y>> ))
```

ggplot()

- ggplot() crea un sistema de coordenadas al que va agregando capas

```
ggplot()
```

- Evidentemente, está vacío: hay que añadir capas
- El primer argumento es el data frame a utilizar
- Podemos omitir el data= por solo poner el df

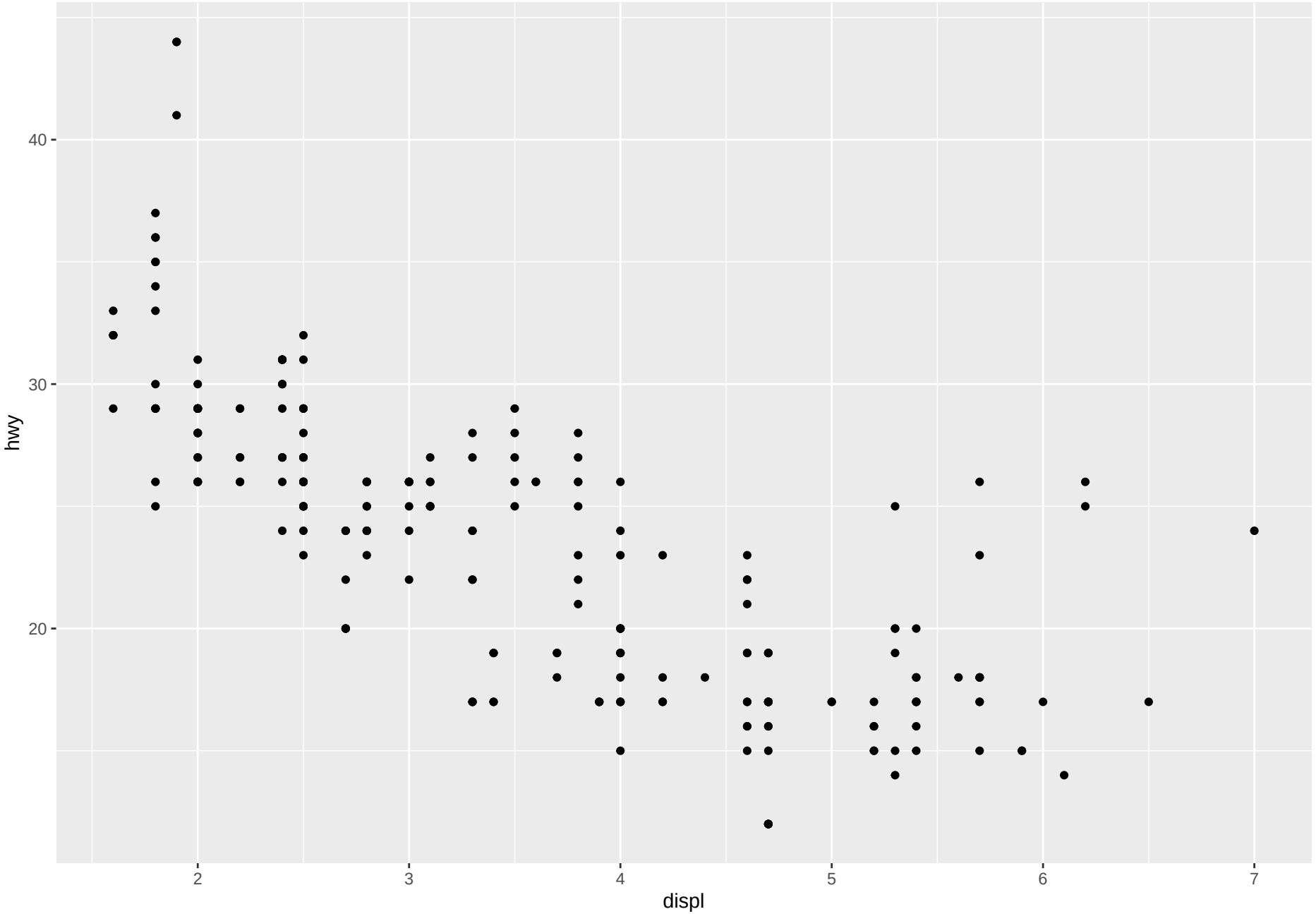
Geometría

- Podemos añadir una geometría en función del tipo de gráfico que queremos
- Define cómo las variables se van a mapear a una visualización
- i.e geom_point() añadiría un gráfico de puntos; geom_line() una línea, etc
- todas las geometrías incluyen un argumento de mapeo, aes(), que especifica qué variables son las que se van a mapear

Ejemplo

- Intentemos graficar la relacion que existe entre el tamaño del motor (disp) y la eficiencia(hwy)

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


40
30
20
hwy
2
3
4
5
6
7
displ

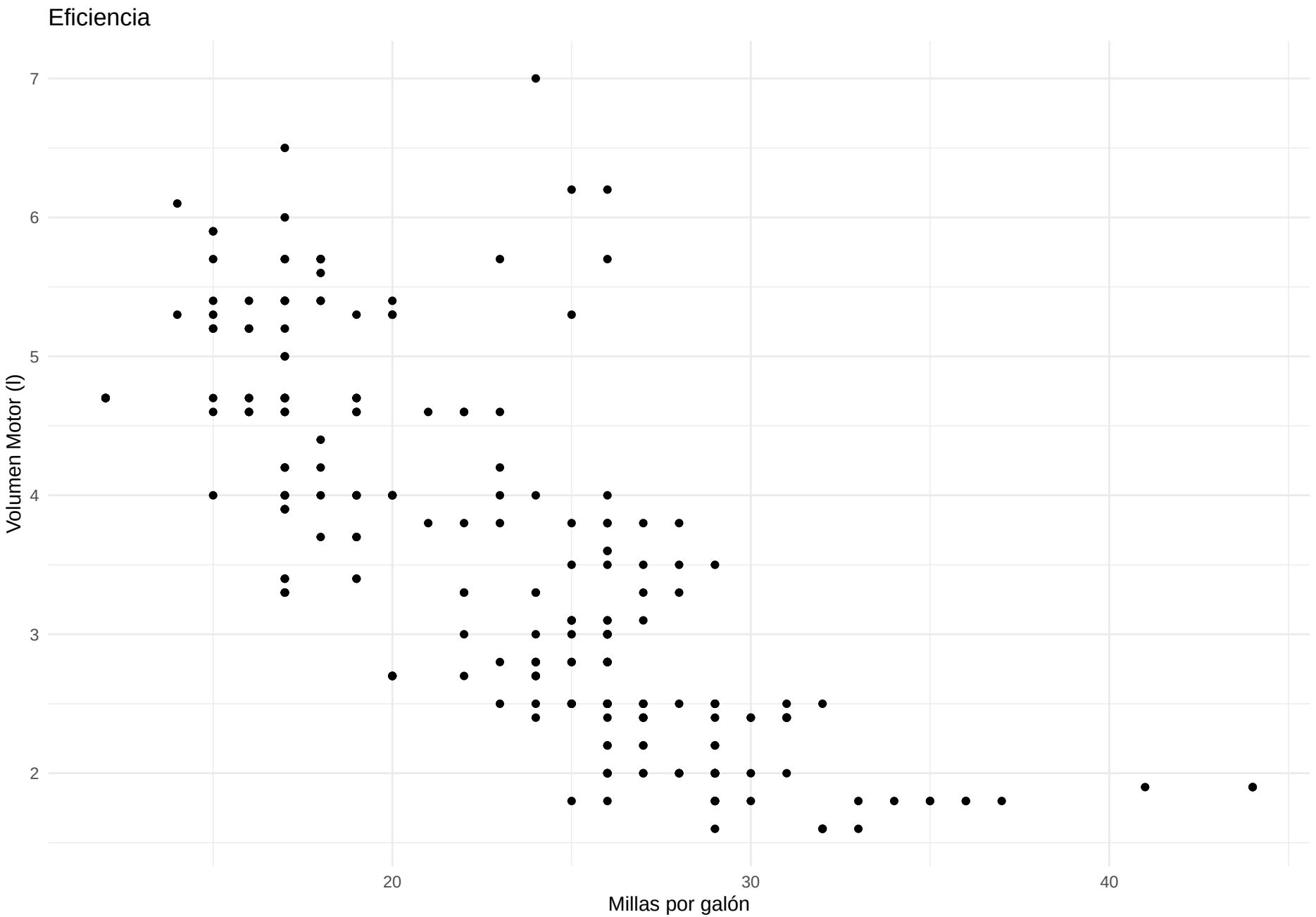
- Podemos guardar las gráficas como objetos, solo que no las mostrará
- Si queremos que lo muestre, podemos encerrar el objeto entre paréntesis al momento de guardarlo

```
(figura.1<- ggplot(data=cars)+
  geom_point(mapping = aes(x= displ, y= hwy)))
```

Añadir elementos

- Podemos añadir elementos con un + elemento()
- Por ejemplo, si queremos añadir etiquetas para los ejes

```
ggplot(data=cars)+
  geom_point(mapping = aes(x= displ, y= hwy))+
  xlab('Volumen Motor (l)')+
  ylab('Millas por galón')+
  ggtitle('Eficiencia')+
  theme_minimal()+
  coord_flip()
```


Eficiencia
Volumen Motor (l)
7
6
5
4
3
2
20
30
40
Millas por galón

- Si ya tenemos la gráfica como objeto guardado, basta con poner el nombre del objeto seguido de los elementos

```
(figura.1+
    xlab('Volumen Motor (l)')+
    ylab('Millas por galón')+
    ggtitle('Eficiencia')+
    theme_minimal()+
    coord_flip())
```

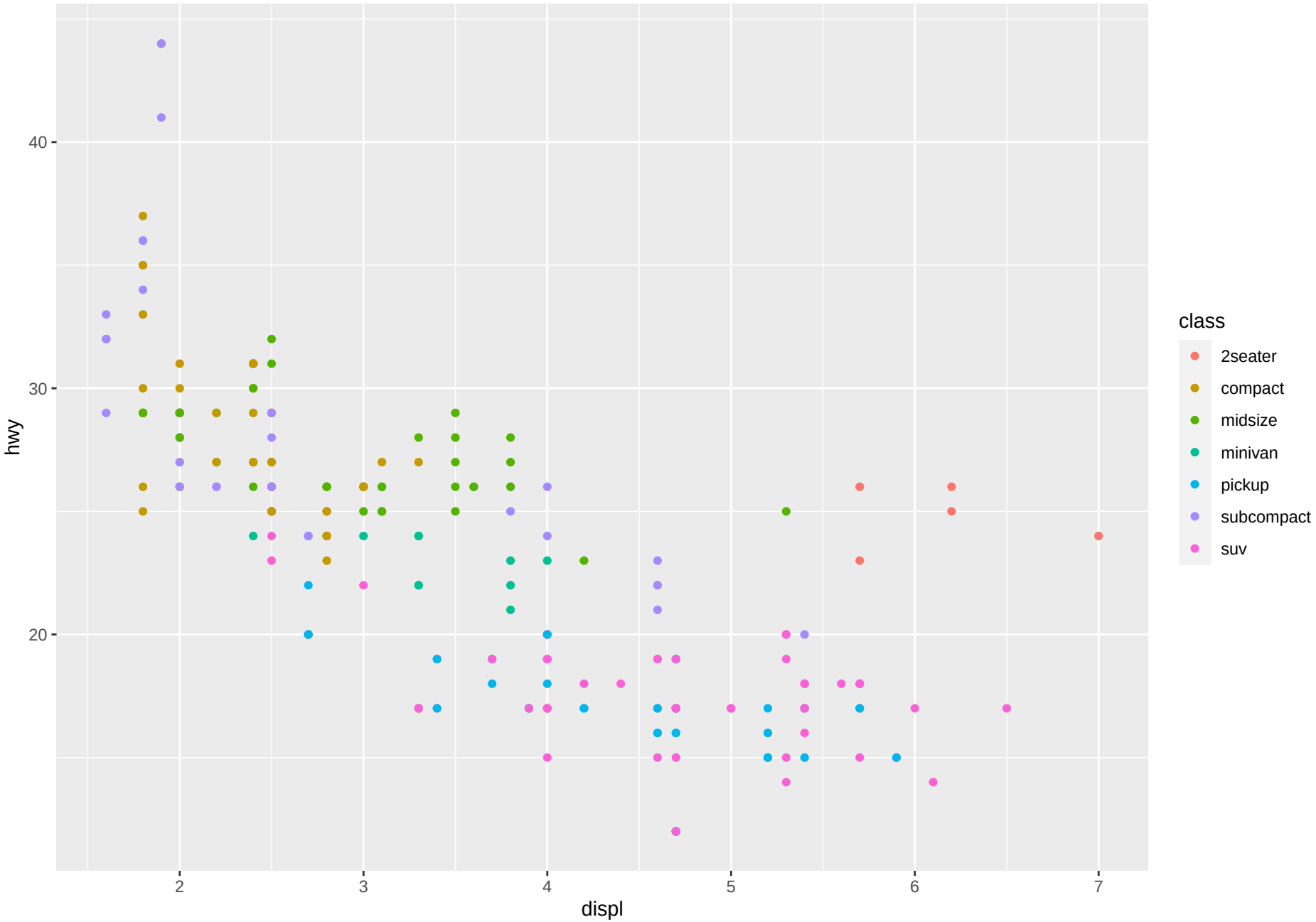
Section 3

Mapeos aestéticos

Mapeos aestéticos

- Una de las virtudes de ggplot es poder vincular una característica estética a una variable
- Por ejemplo, podríamos vincular el tipo de vehículo con la forma y color de los puntos

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

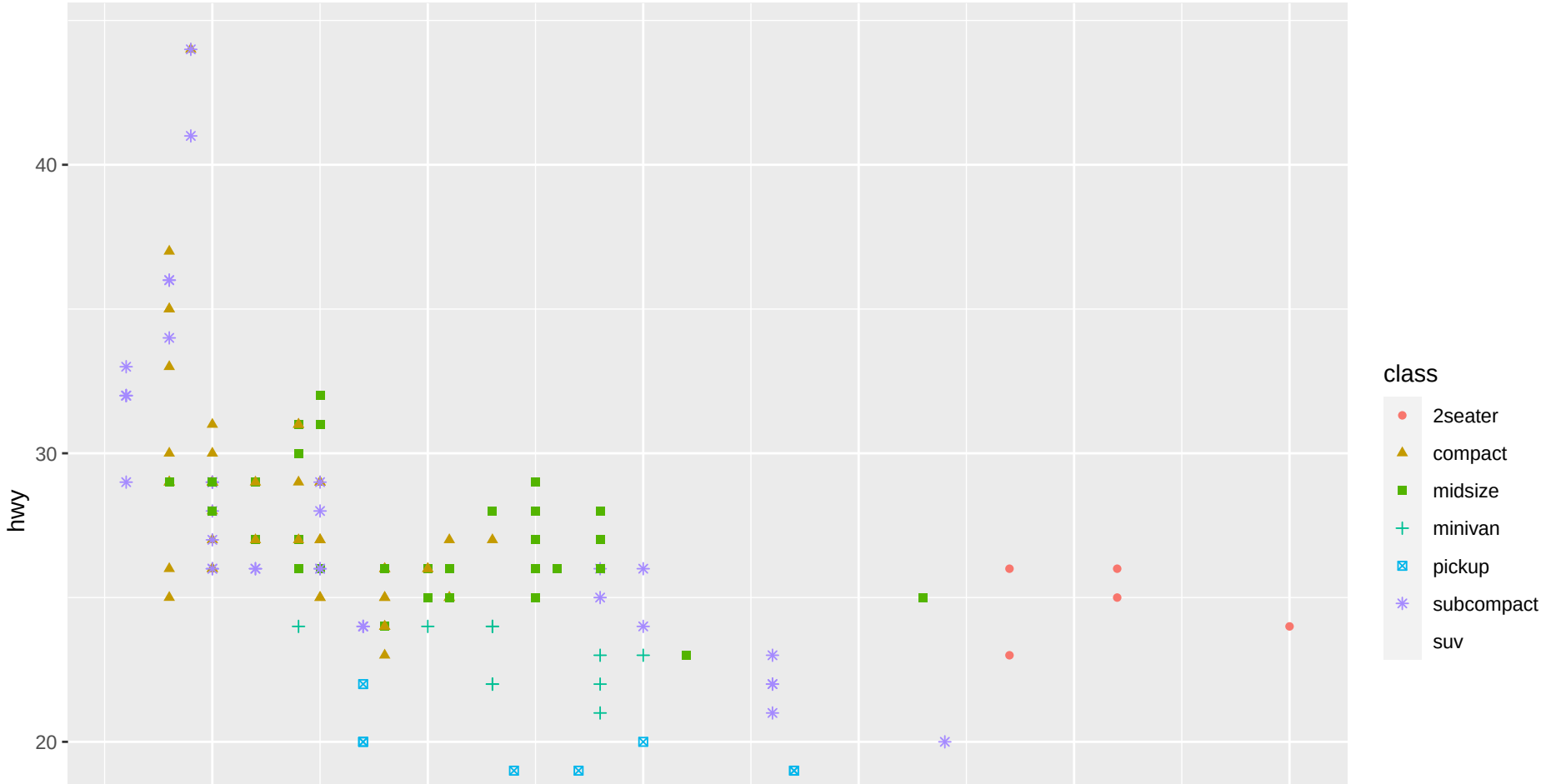

hwy
40
30
20
2
3
4
5
6
7
displ
class
2seater
compact
midsize
minivan
pickup
subcompact
suv

- Lo que hace ggplot es asignar un nivel de color a cada categoría de coches
- El color asignado da información de la variable clase
- ggplot automáticamente añade una leyenda que muestra los niveles asignados

- Intentemos añadir formas

```
(aes.2<- ggplot(data=cars)+
  geom_point(mapping = aes(x= displ, y= hwy,
                           color=class, shape=class)))
```

```
## Warning: The shape palette can deal with a maximum of 6 dis
## more than 6 becomes difficult to discriminate; you have 7.
## specifying shapes manually if you must have them.

## Warning: Removed 62 rows containing missing values (geom_po
```

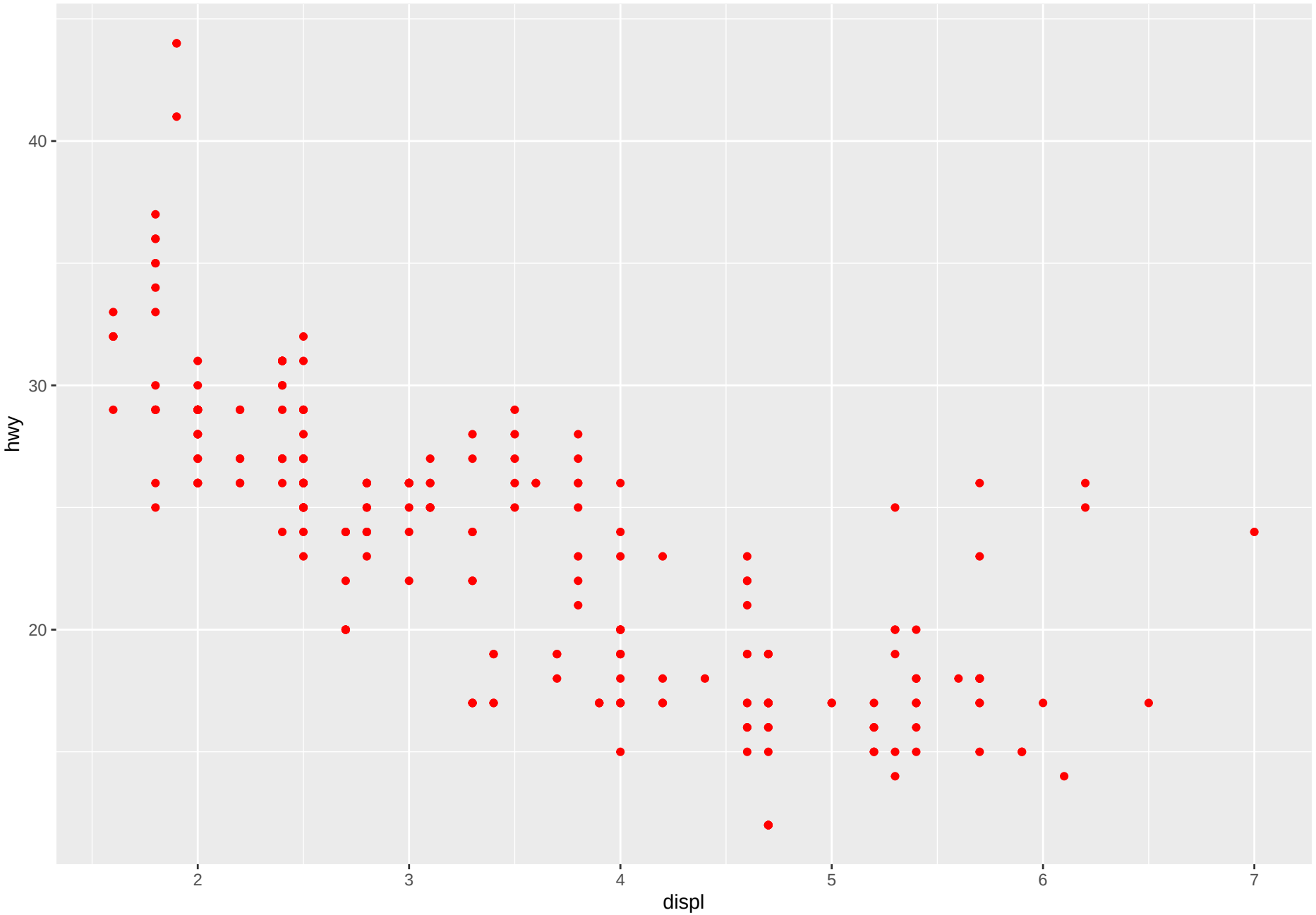


- Por default, ggplot tiene un número limitado de formas
- Omite la categoría para la que no alcanzó forma

Dentro vs fuera de aes()

- Podemos poner color, forma , etc. fuera de la aestética
- Al hacerlo, estamos especificando a ggplot que queremos esa geometría en particular del color especificado

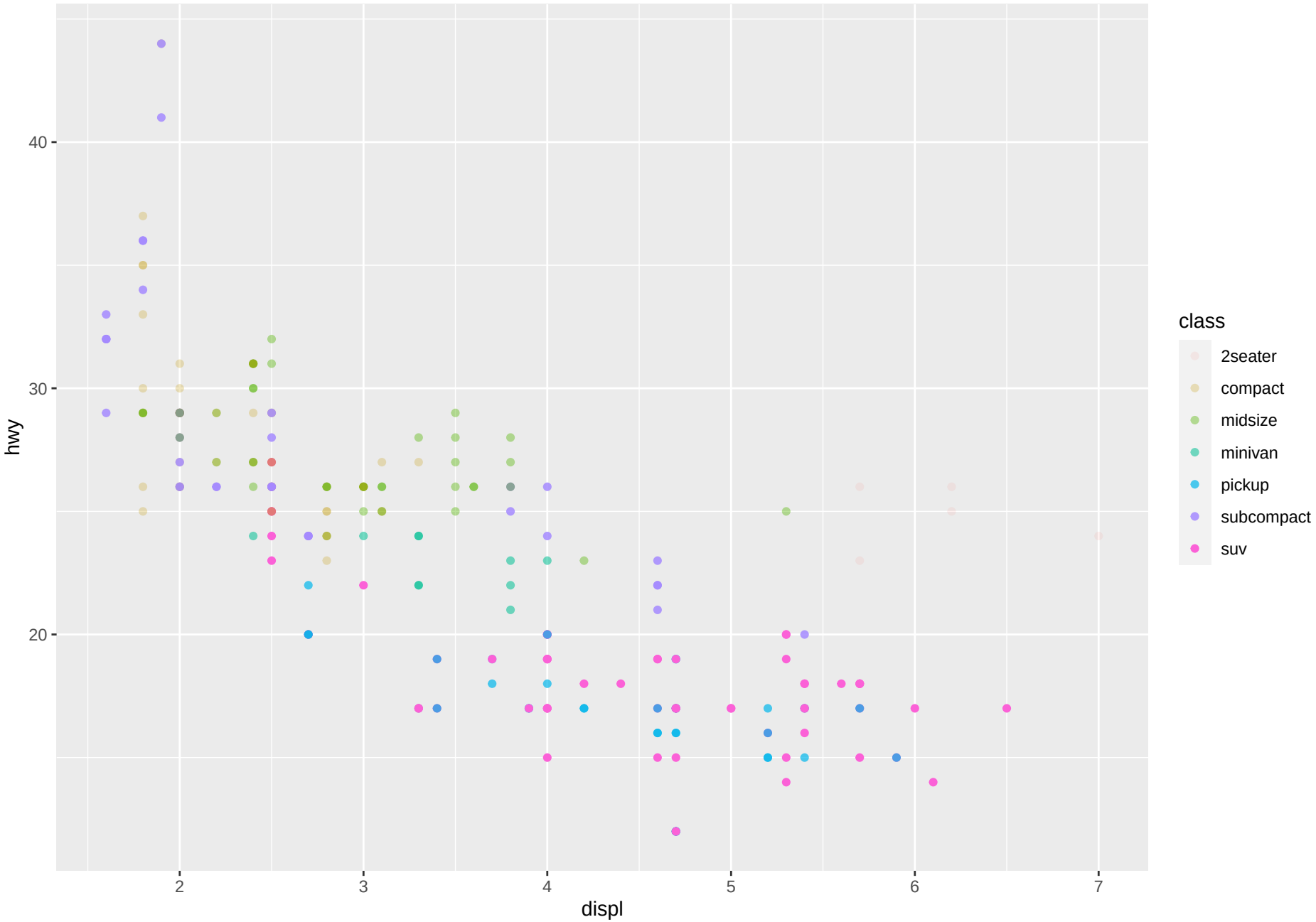
```
(fuera_aes<- ggplot(data=cars)+
  geom_point(mapping = aes(x= displ, y= hwy) ,
                           color= 'red'))
```


hwy
40
30
20
2
3
4
5
6
7
displ

alpha

- Podemos modificar la transparencia usando la aestética alpha
- Si lo ponemos fuera de aes() podemos especificar la transparencia de la geometría

```
(aes.3<- ggplot(data=cars)+
    geom_point(mapping = aes(x= displ, y= hwy,
                             color=class, alpha=class)))
```


hwy
40
30
20
2
3
4
5
6
7
displ
class
2seater
compact
midsize
minivan
pickup
subcompact
suv

Section 4

Escalas de colores

Escalas de colores

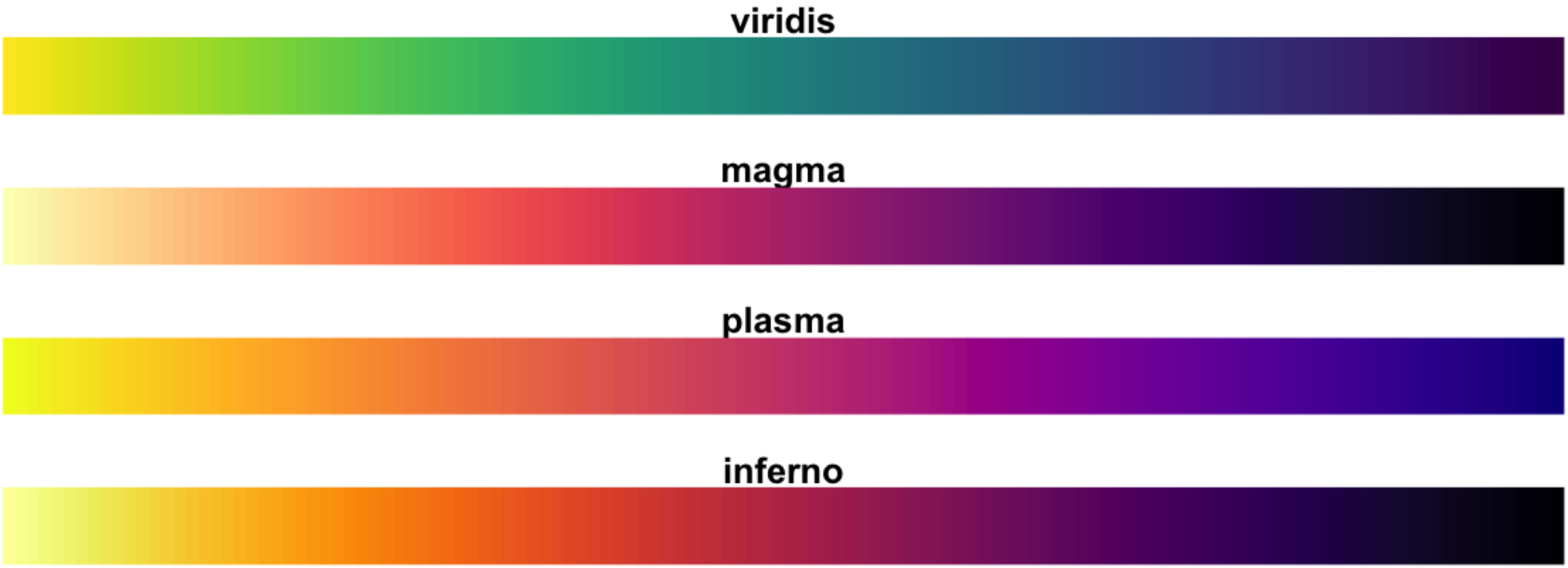
- Añadiendo un elemento podemos escoger la paleta de colores
- Existen paquetes que con distintas paletas
- Al final , una variable (categórica o continua) va a estar vinculada a una paleta de nuestra elección

- Probemos con Viridis

```
library(viridis)
```

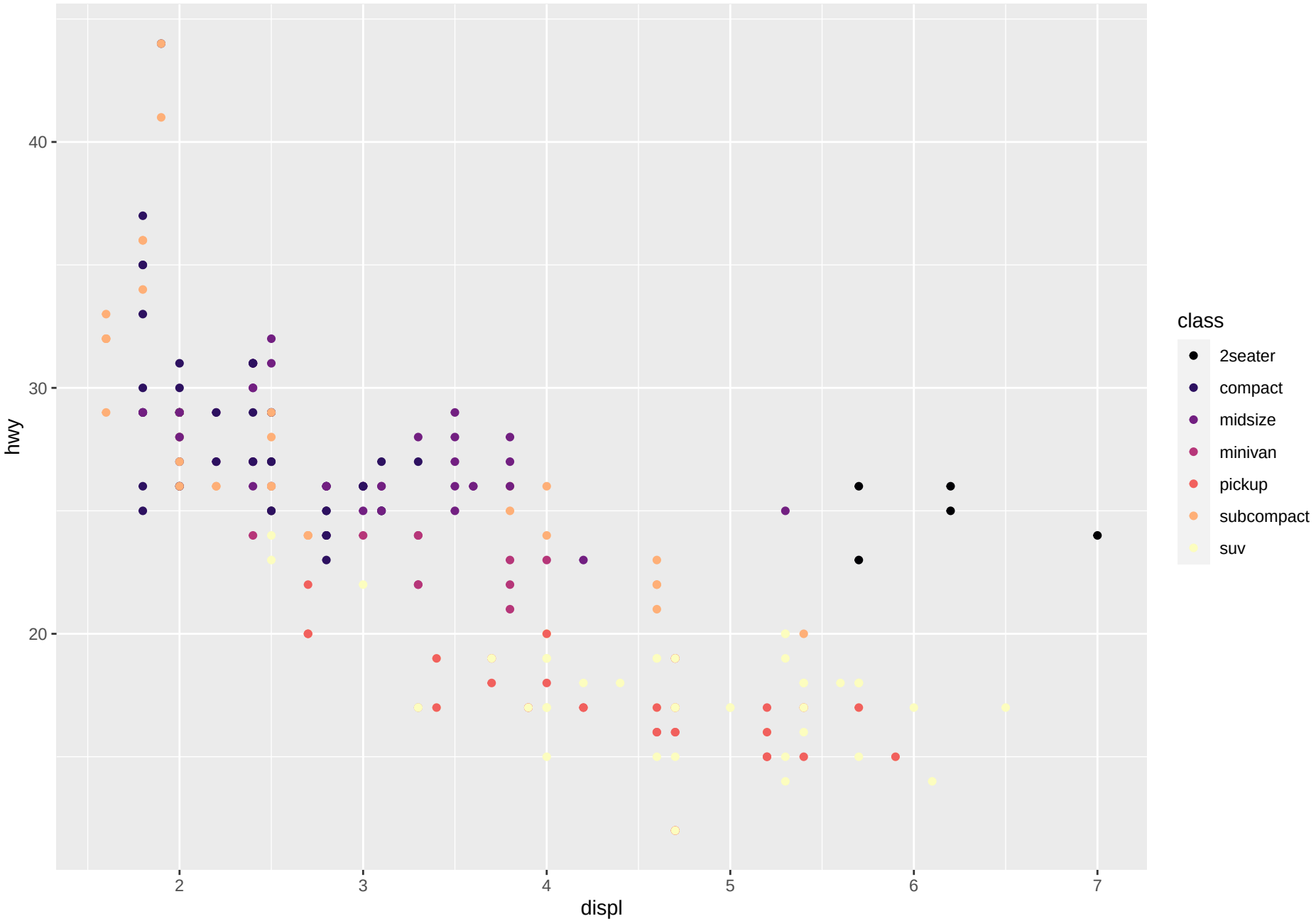
```
## Loading required package: viridisLite
```

- Todos los paquetes de paletas tienen múltiples gamas de colores a escoger
- Tendremos que especificar cual queremos


viridis
magma
plasma
inferno

- Repitamos nuestra primera gráfica con aestética de color, pero usando la paleta viridis

```
(scale_color <- ggplot(data=cars)+
    geom_point(mapping = aes(x= displ, y= hwy,
                             color=class))+
    scale_color_viridis(discrete = T,
                        option = 'magma'))
```


hwy
40
30
20
2
3
4
5
6
7
displ
class
2seater
compact
midsize
minivan
pickup
subcompact
suv

Section 5

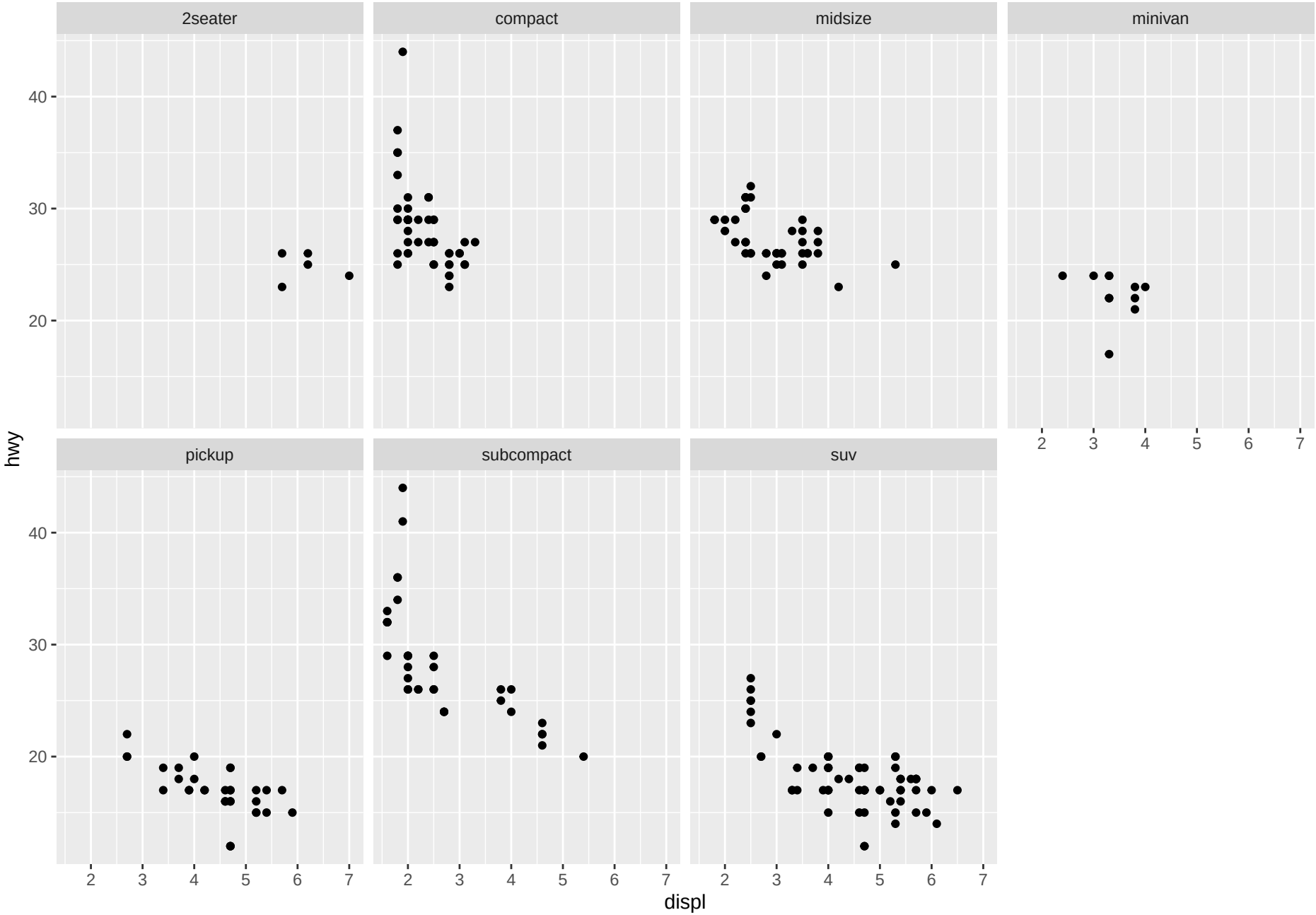
Facets

Facets

- Usando la aestética podemos mostrar en un mismo gráfico figuras para cada categoría
- Los facets sirven para mostrar una gráfica distinta para cada categoría
- Tratemos de hacer lo mismo de los colores, pero con facets

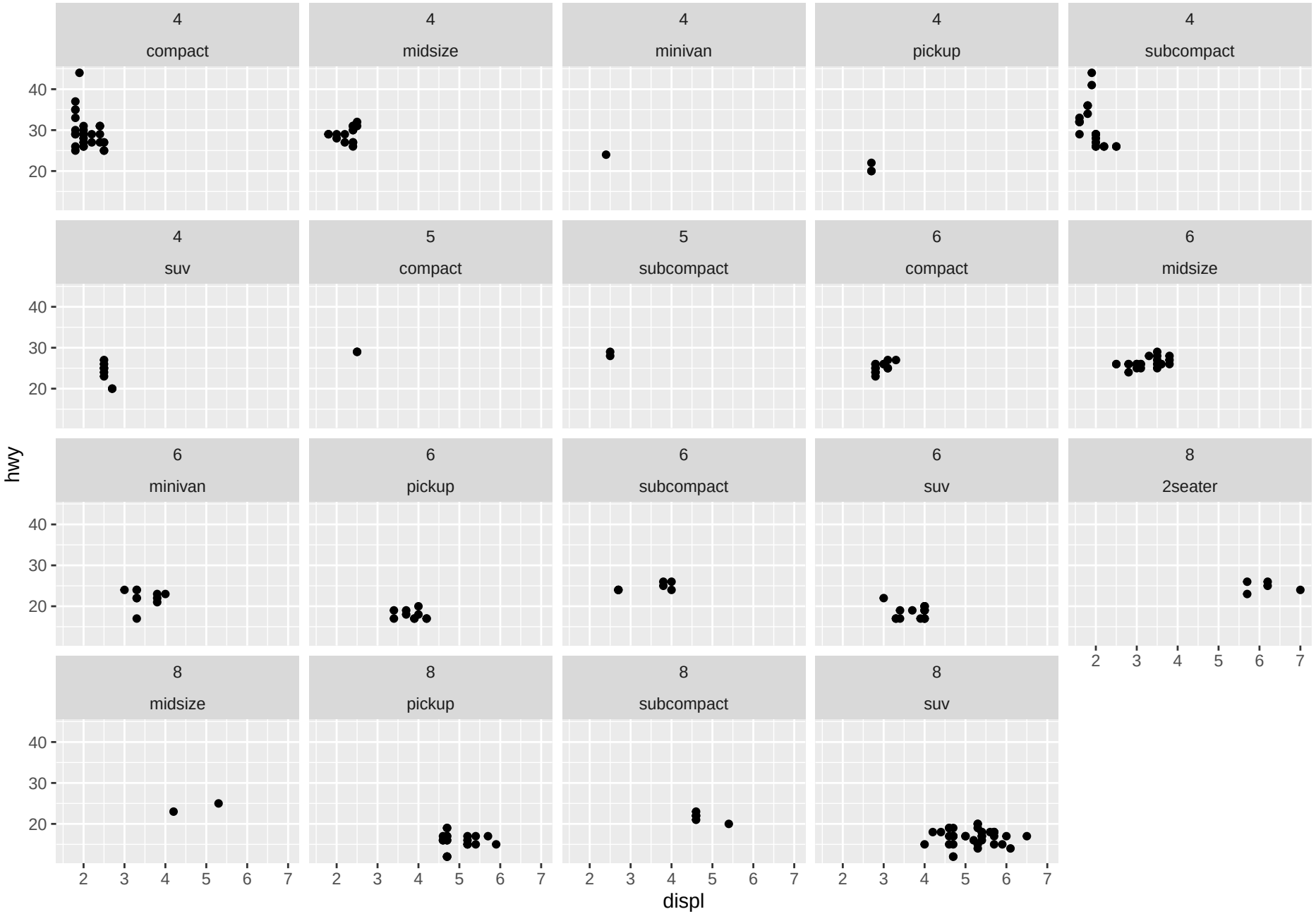
```
(facets <- ggplot(cars)+
    geom_point(mapping =
                  aes(x= displ, y= hwy))+
    facet_wrap(~ class, nrow = 2))
```

- Debemos incluir la fórmula empezando por $\sim$
  - La fórmula es la variable por la que vamos a descomponer las gráricas


2seater
compact
midsize
minivan
pickup
subcompact
suv
hwy
displ
40
30
20
2
3
4
5
6
7

- Podemos hacer facets por más variables, cambiando la fórmula

```
(facets <- ggplot(cars)+
    geom_point(mapping =
                  aes(x= displ, y= hwy))+
    facet_wrap(cyl ~ class))
```


4 compact
4 midsize
4 minivan
4 pickup
4 subcompact
4 suv
5 compact
5 subcompact
6 compact
6 midsize
6 minivan
6 pickup
6 subcompact
6 suv
8 2seater
8 midsize
8 pickup
8 subcompact
8 suv
hwy
displ
40
30
20
2
3
4
5
6
7

Section 6

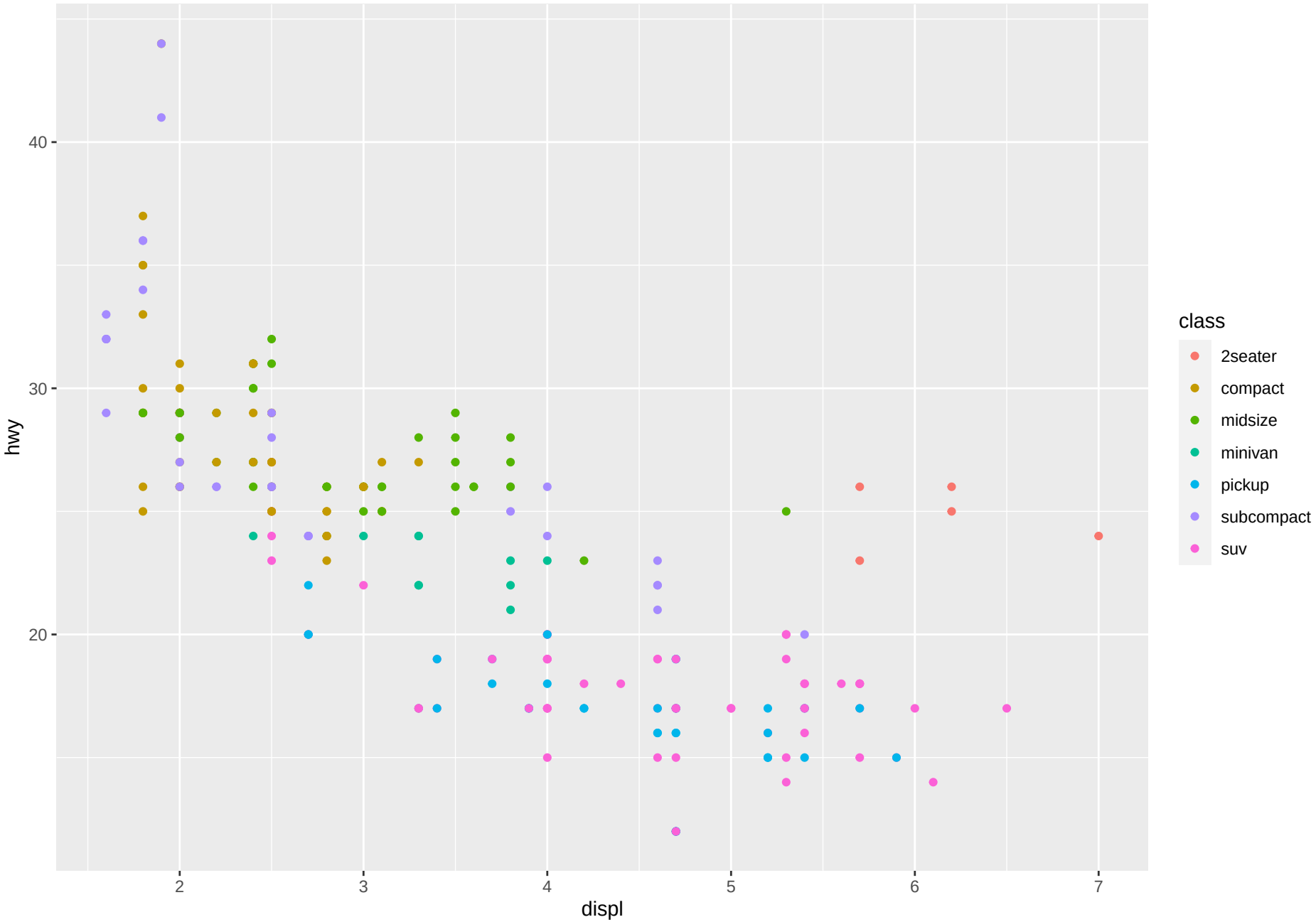
Otra sintaxis

Otra sintaxis

- ggplot admite otra sintáxis que puede resultar más fácil
- Además, podemos omitir pedazos de los argumentos para que las gráficas tengan menos texto
- En particular, podemos incluir la estética dentro de ggplot() y dejar la geometría vacía

```
(figura.1.2 <- ggplot(cars,
                      aes(displ, hwy, color= class))+
   geom_point())
```

- Podemos omitir la x,y y el mapeo
- dejamos la geometría vacía y ponemos el mapeo en ggplot()
- Argumetos adicionales de la geometría se siguen especificando dentro (como la transparencia, forma o tipo de línea o punto)


hwy
40
30
20
2
3
4
5
6
7
displ
class
2seater
compact
midsize
minivan
pickup
subcompact
suv

Section 7

Múltiples geometrías

Variedad de gráficos

- Existe una gran variedad de geometrías
- Si bien se añaden de la misma manera, requieren de argumentos o fórmulas distintas
- Veamos algunas, además de puntos y líneas:
  - Línea de regresión
  - Gráfico de barras
  - Histograma
  - Densidad

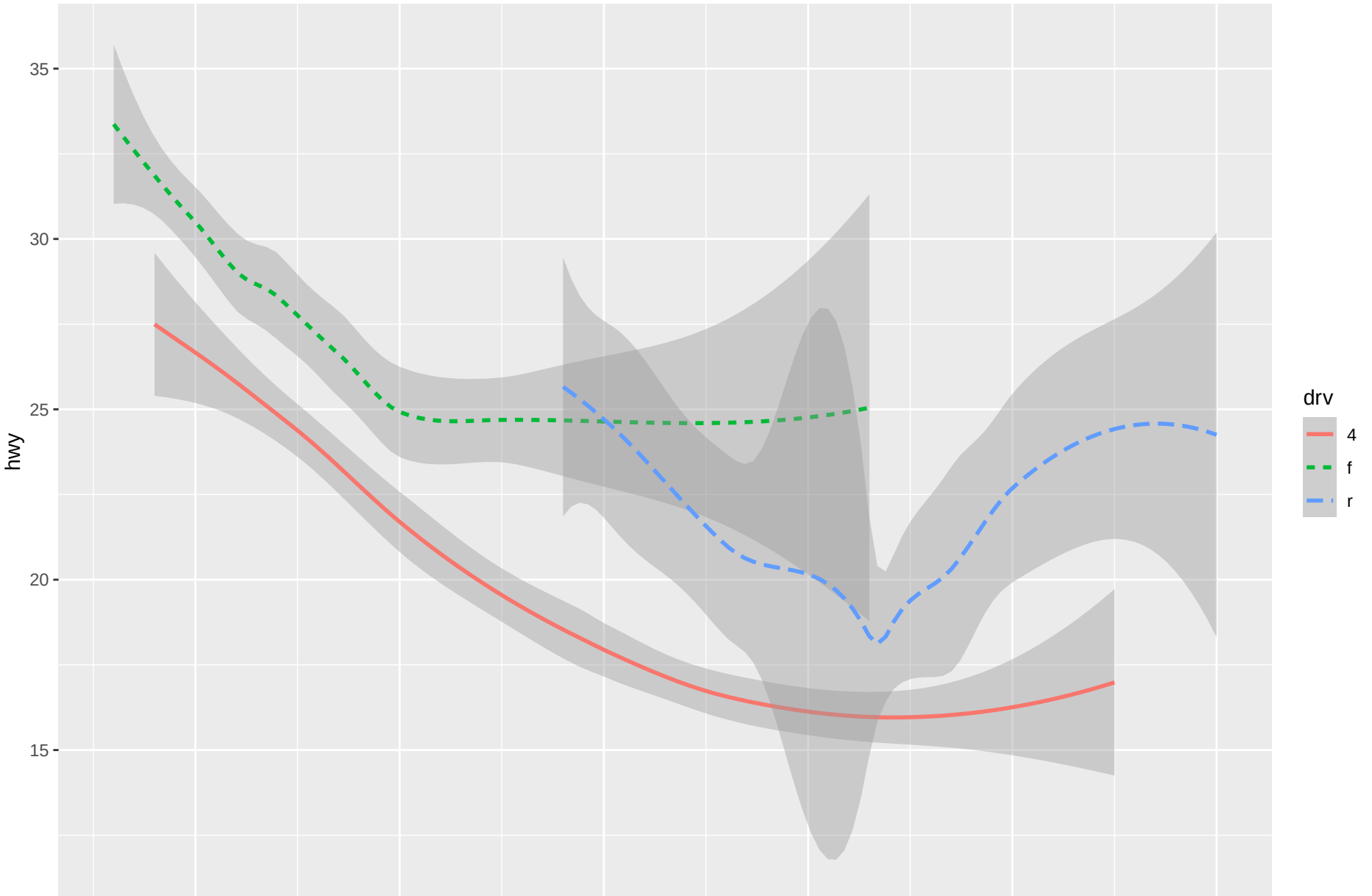
geom_smooth()

- geom_smooth() añade la línea de regresión de las variables especificada.
- Automáticamente añade intervalos de confianza

- Hagamos lo mismo, con una línea para cada tipo de tracción (drv), pues son menos categorías

```
(smooth <- ggplot(cars)+
    geom_smooth(mapping = aes(
      x=displ, y= hwy,
      linetype= drv, color=drv
    )))
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

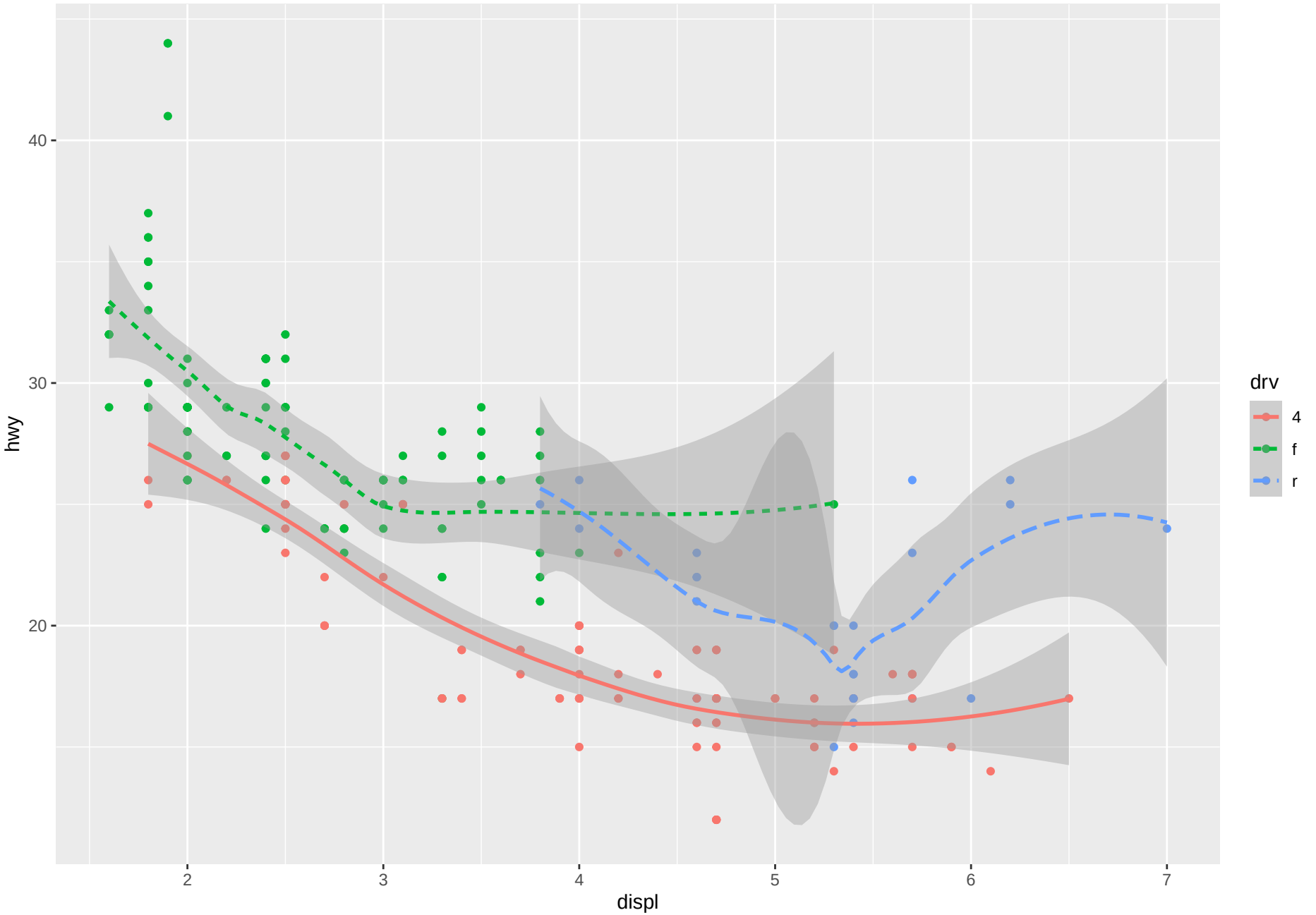


Múltiples geometrías

- Es común incluir las observaciones y la línea de regresión en el mismo gráfico
- Si utilizamos la sintaxis alternativa, es muy sencillo añadir múltiples geometrías

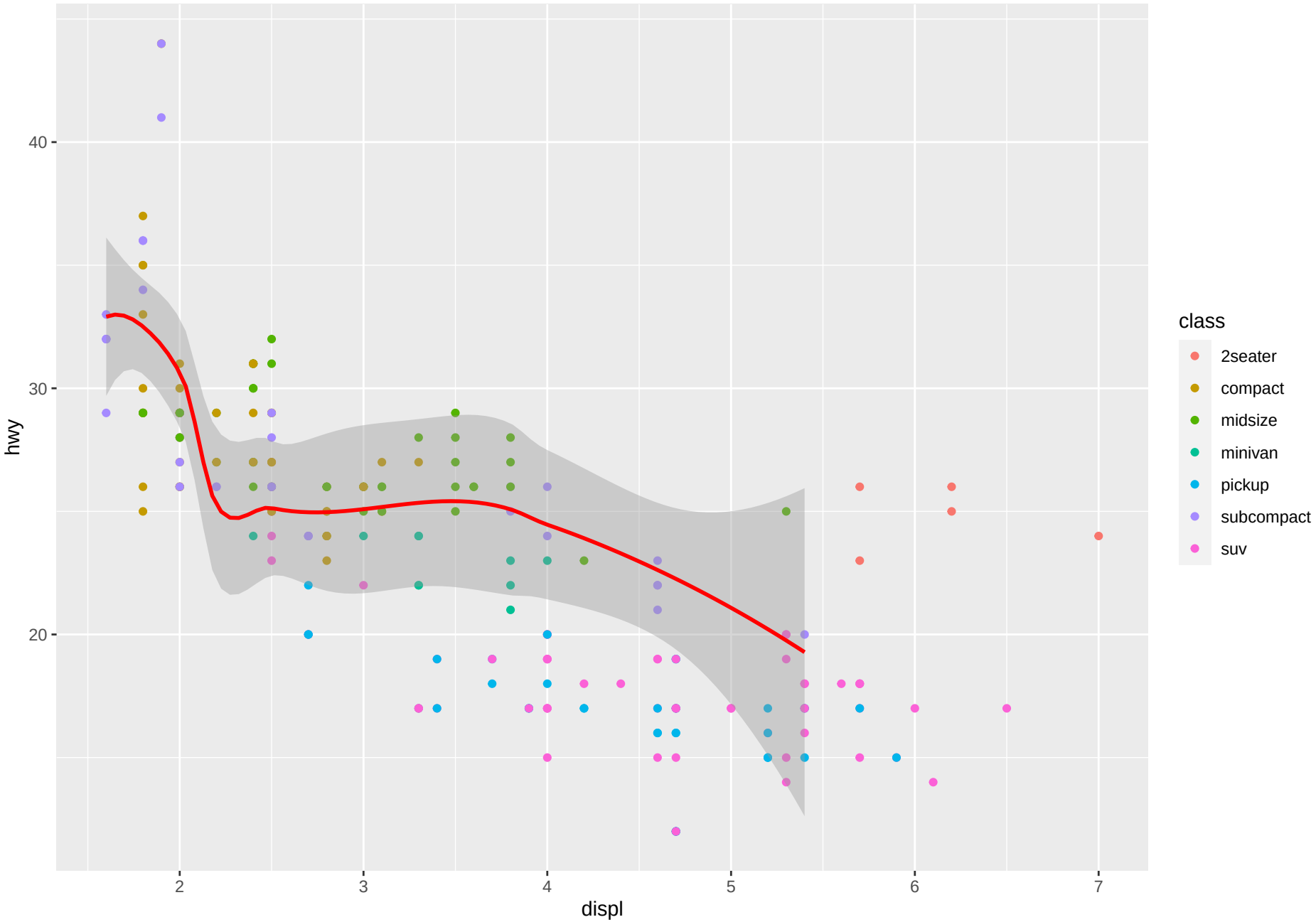
```
(mult_geom <- ggplot(cars,
                     aes(displ, hwy,
                         color= drv,
                         linetype=drv))+
    geom_point()+
    geom_smooth())
```

- Lo común va dentro de ggplot() y lo que es propio de cada geometría va dentro de su respectiva geom


hwy
40
30
20
2
3
4
5
6
7
displ
drv
4
f
r

- Podemos jugar con las configuraciones

```
(mult_geom.2 <- ggplot(mpg,
                       aes(displ, hwy))+
    geom_point(aes(color=class))+
    geom_smooth(data= filter(mpg,
                             class=='subcompact'),
                color= 'red'))
```


hwy
40
30
20
2
3
4
5
6
7
displ
class
2seater
compact
midsize
minivan
pickup
subcompact
suv

- Al incluir x y y en ggplot, compartirán las mismas variables
- Como color solo está en la aestética de point, solo los puntos varían de color de acuerdo con la categoría
- Usamos un subconjunto de datos para graficar la línea de regresión
- La línea es roja porque no está en la aestética y solo está en la geometría de la regresión

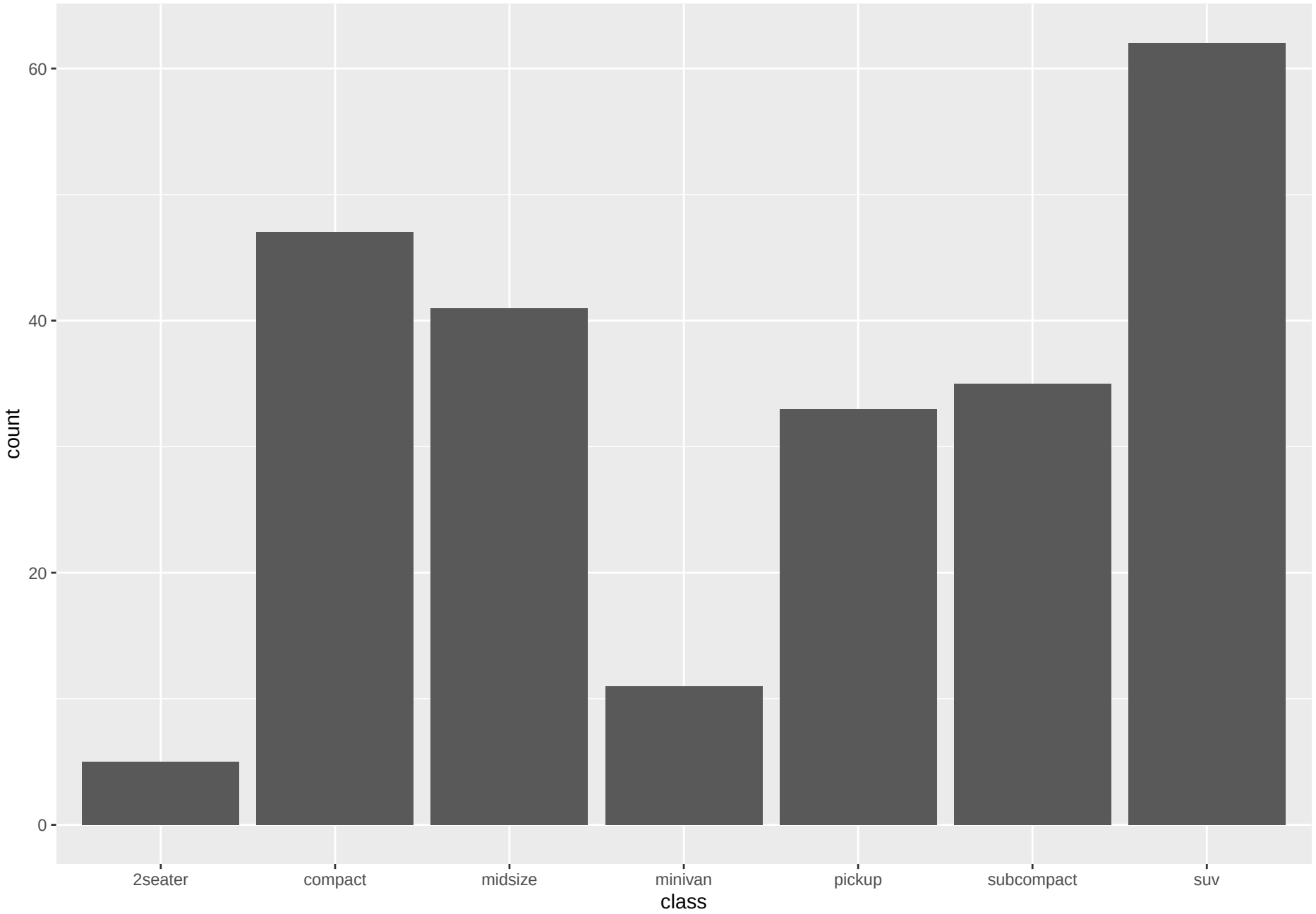
Section 8

Gráficos de barras

Gráfico de barras

- Solo requerimos cambiar la geometría

```
(bar <- ggplot(cars)+
  geom_bar(aes(class)))
```


60
40
20
0
count
2seater
compact
midsize
minivan
pickup
subcompact
suv
class

- Construyamos un df que haga más fácil ver lo que hace el gráfico

```
cars_count <- cars %>%
  group_by(class) %>%
  summarise(n=n())
```

- ¿Cuál será el output?

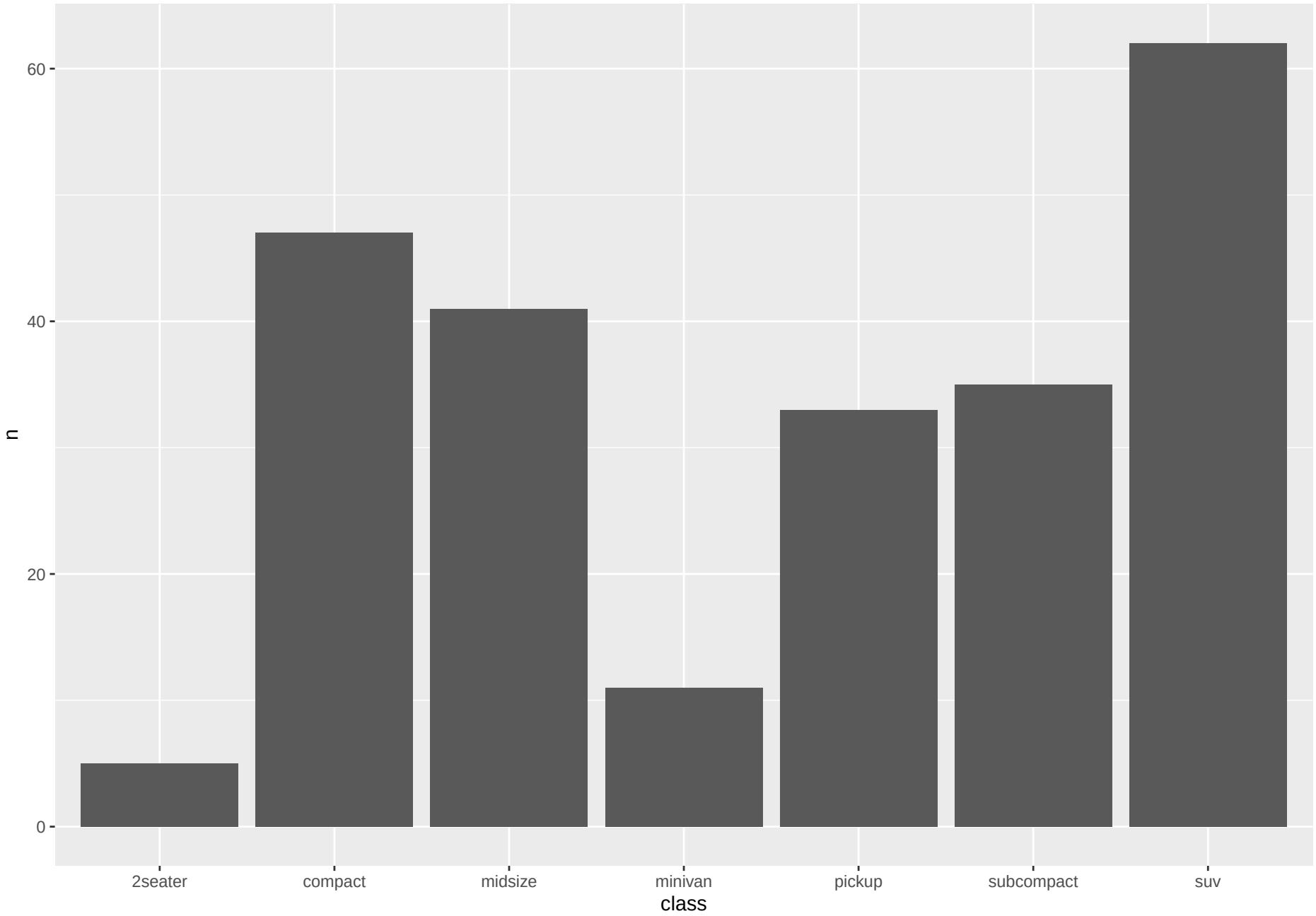
	class	n
1	2seater	5
2	compact	47
3	midsize	41
4	minivan	11
5	pickup	33
6	subcompact	35
7	suv	62

- ¿Por qué una sola variable?
- Los gráficos de barras cuentan, pero esa cuenta no está explícita en el df, sino que ggplot lo calculó (e hizo algo análogo al df que creamos)
- El algoritmo que calcula un nuevo valor para la gráfica se llama stat
- Al hacer una gráfica de barras de esta manera, ggplot calcula dos nuevos valores, la cuenta y la proporcion
- El stat default para geom_bar() es 'count', de ahí que al solo poner la variable 'class' nos entregó la cuenta de las observaciones de cada clase

Identity

- Podemos especificar el stat que queremos
- Por ejemplo, para hacer un gráfico de barras, sin que calcule otra cosa, usamos el stat=‘identity’
- Tomemos el df que hicimos

```
(cars_count <- cars %>%
  group_by(class) %>%
  summarise(n=n()) %>%
  ggplot(aes(class, n))+
  geom_bar(stat = 'identity'))
```


60
40
20
0
n
2seater
compact
midsize
minivan
pickup
subcompact
suv
class

stat='identity'

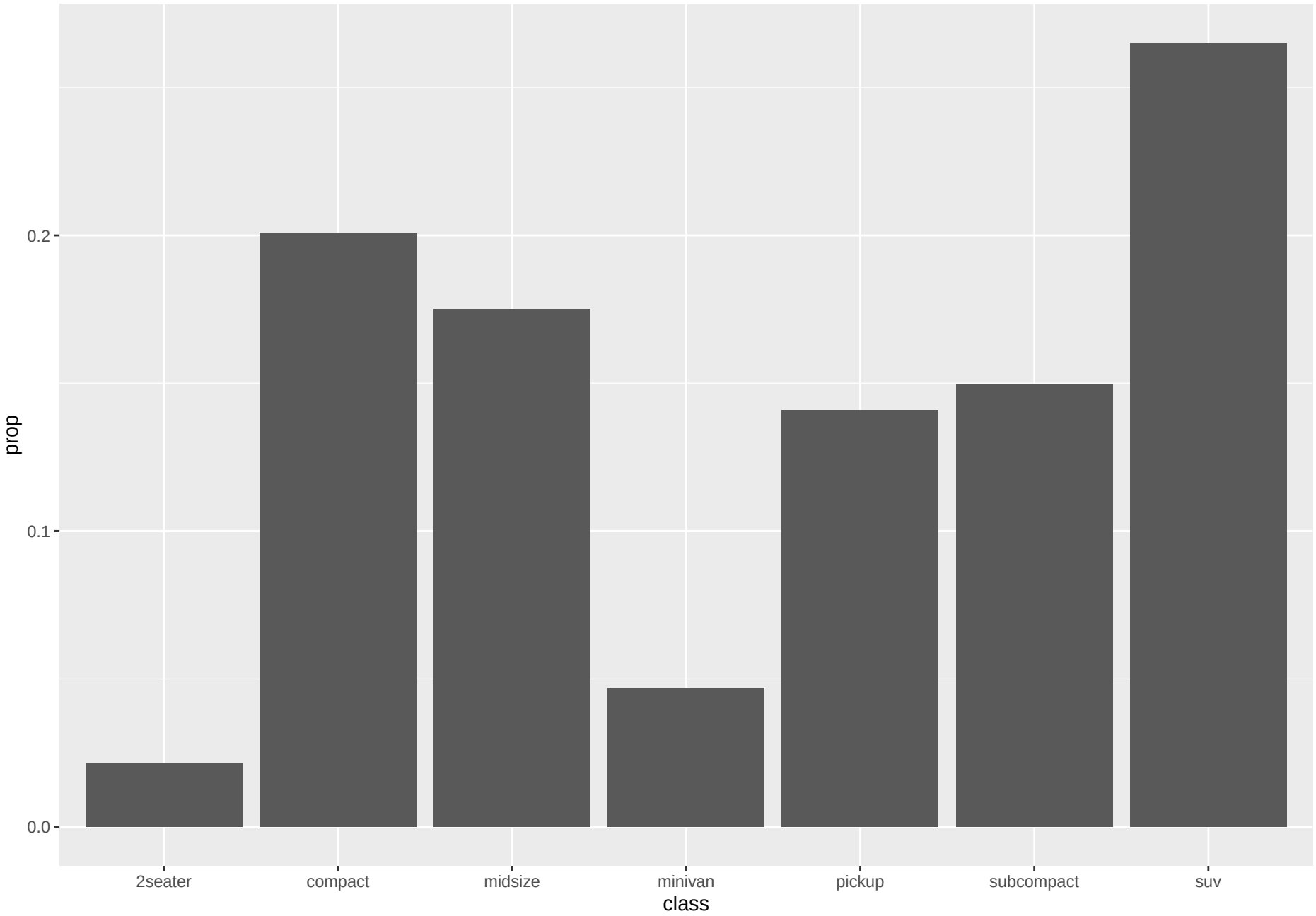
- Así, hacemos un gráfico tradicional: vinculamos una variable x a una y
- ggplot no calculó nada, sino que solo graficó dos variables preexistentes

Barras de proporciones

- Para hacer un gráfico de barras, pero con porcentajes, tenemos que cambiar un poco el gráfico

```
(bar_prop <- ggplot(cars)+
    geom_bar(aes(class, y=..prop..,
                 group=1)))
```

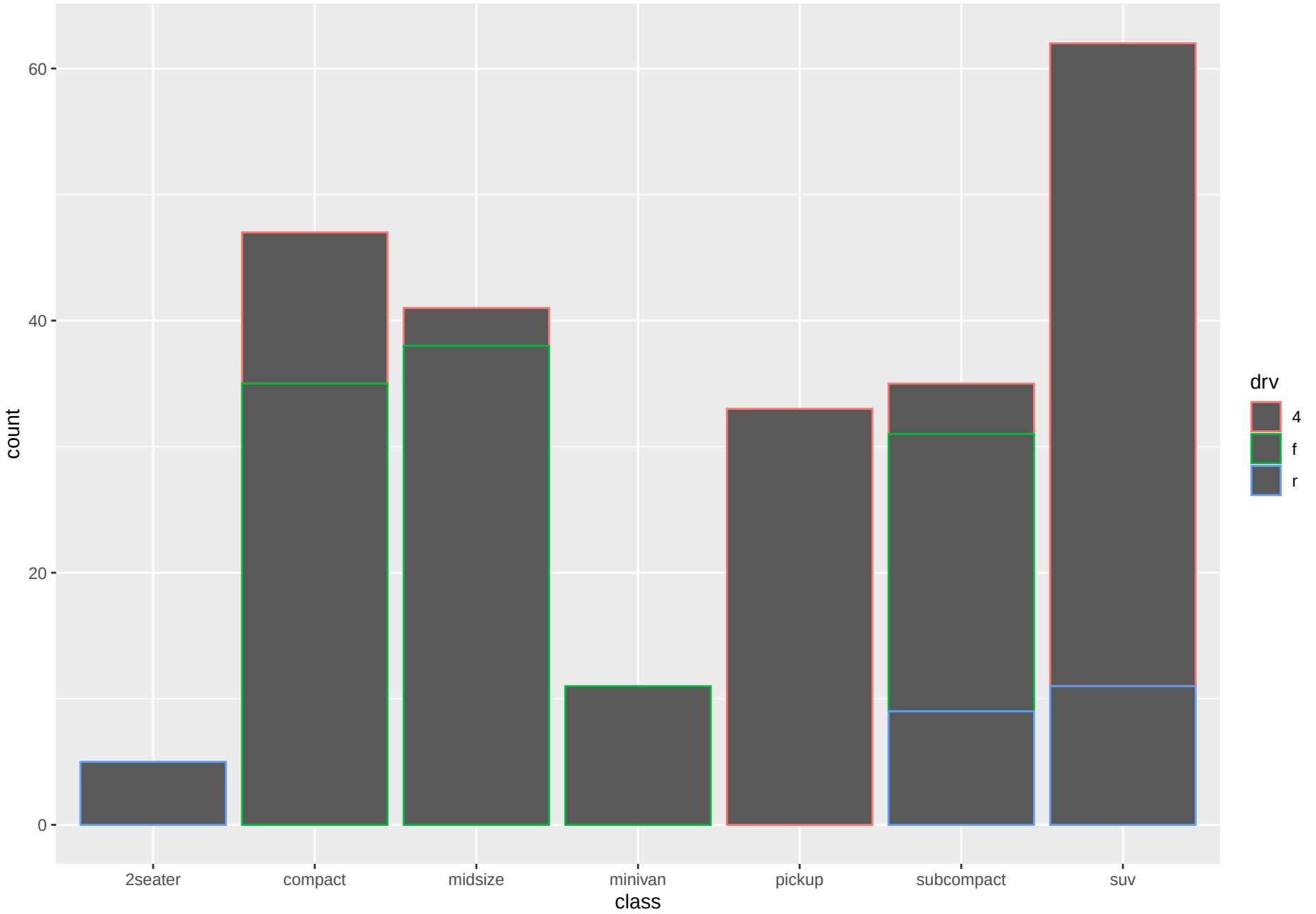
- ¿Qué pasa si no pongo la variable grupo?


0.2
0.1
0.0
prop
2seater
compact
midsize
minivan
pickup
subcompact
suv
class

fill vs color

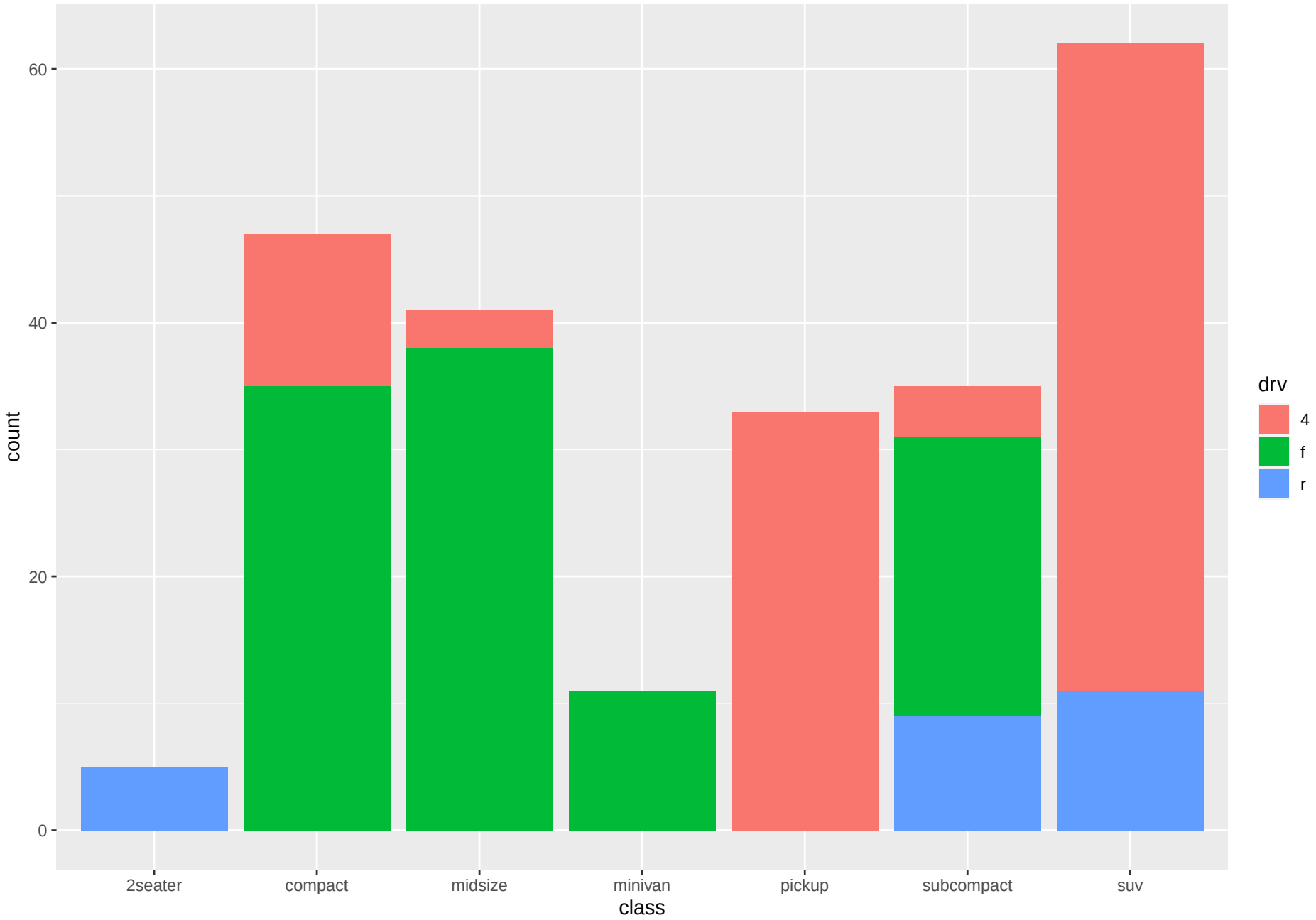
- Así como con los gráficos pasados, podemos vincular el color a una variable
- Hagamos el mismo gráfico de barras de proporción, solo que, para cada clase, distinguir cuántos coches hay de cada tracción dentro de cada clase

```
(bar_color <- ggplot(cars)+
    geom_bar(aes(class,
                 color=drv)))
```


60
40
20
0
count
2seater
compact
midsize
minivan
pickup
subcompact
suv
class
drv
4
f
r

- En realidad, la aestética color solo modifica el borde
  - ¿Por qué funcionaba con líneas y puntos?
- Si queremos modificar el relleno, necesitamos usar la aestética fill

```
(bar_fill <- ggplot(cars)+
    geom_bar(aes(class,
                 fill=drv)))
```

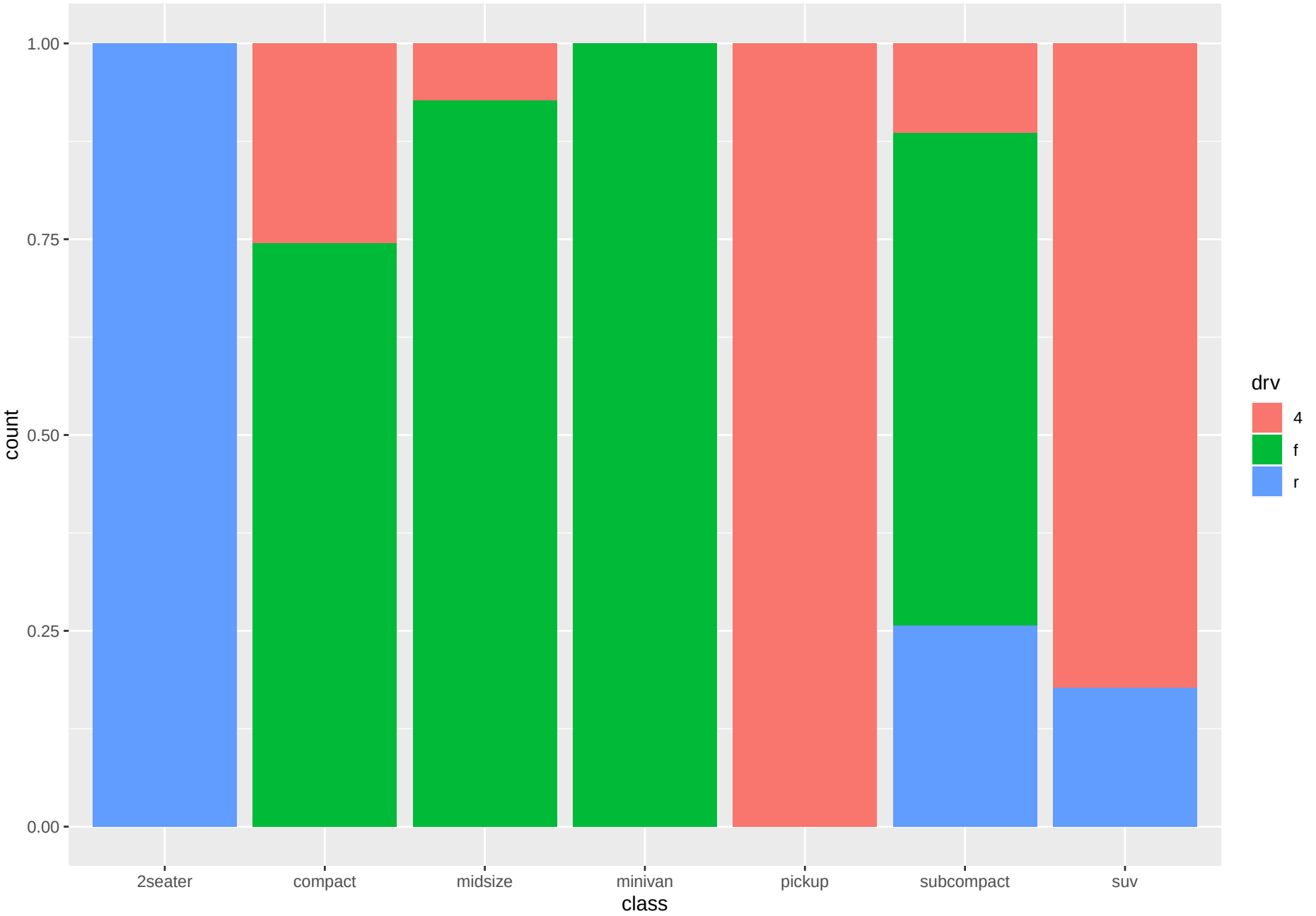

60
40
20
0
count
2seater
compact
midsize
minivan
pickup
subcompact
suv
class
drv
4
f
r

position

- Por default, ggplot apiló las barras unas encima de las otras
- Es que el argumento *position* de la geometría es por default 'stack'
- Podemos cambiar las posiciones de las barras
- otras posiciones son
  - position='fill' las apila, pero hace las barras del mismo tamaño
  - position='dodge' pone las barras una al lado de la otra en pequeños grupos

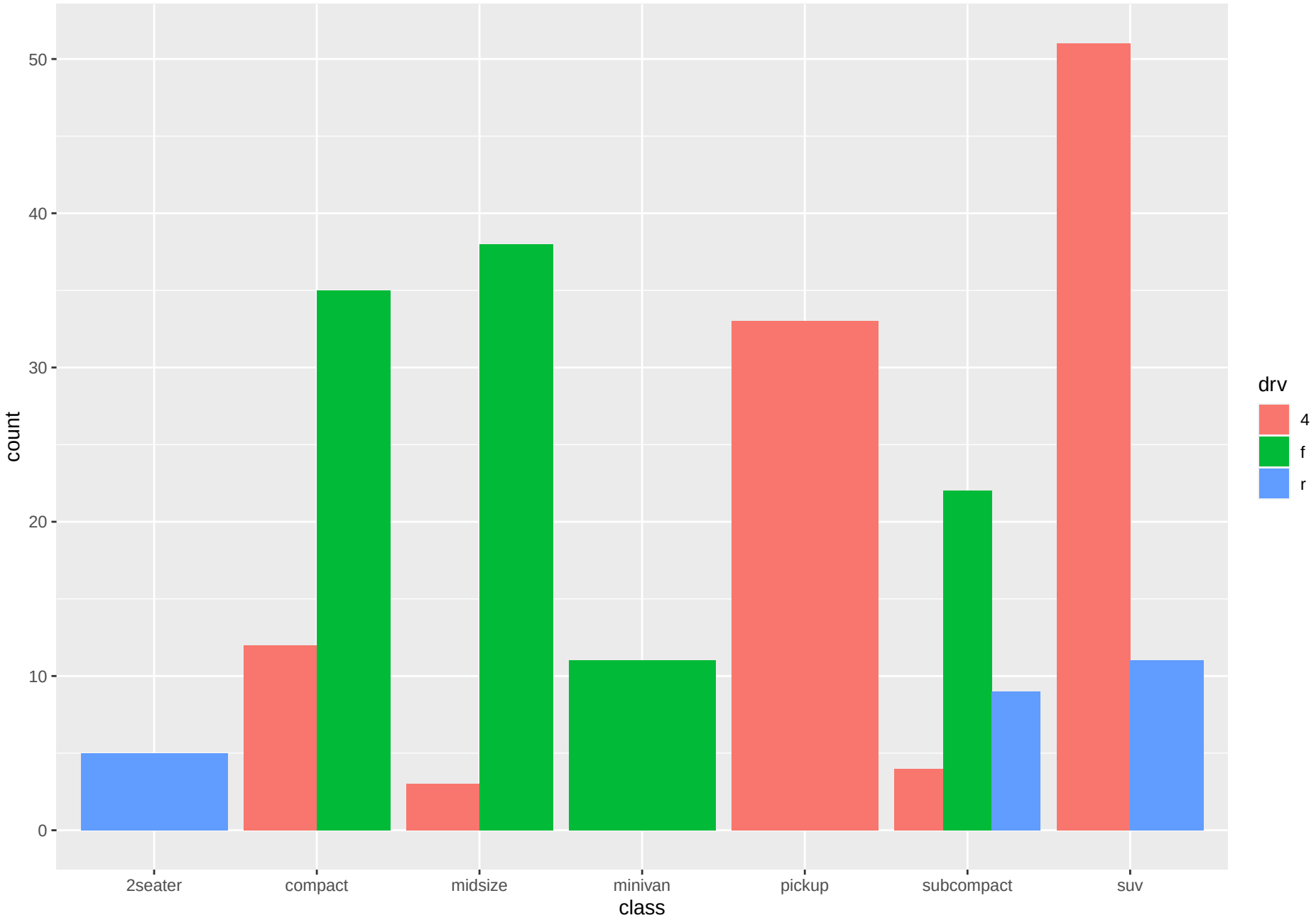
position='fill'

```
(bar_position_fill <- ggplot(cars)+
    geom_bar(aes(class,
                 fill=drv), position='fill'))
```


1.00
0.75
0.50
0.25
0.00
count
2seater
compact
midsize
minivan
pickup
subcompact
suv
class
drv
4
f
r

position='dodge'

```
(bar_position_dodge <- ggplot(cars)+
    geom_bar(aes(class,
                 fill=drv), position='dodge'))
```


50
40
30
20
10
0
count
2seater
compact
midsize
minivan
pickup
subcompact
suv
class
drv
4
f
r

Section 9

Histograma y Densidad

Histograma y Densidad

- Para ilustrar el histograma y la densidad creemos un df de salarios de n trabajadores con
  - Una variable id
  - Una variable que identifique el género
  - su salario
- Supongamos que

$$w_{Hi} \sim N(30000, 3000)$$

$$w_{Mi} \sim N(24000, 5000)$$

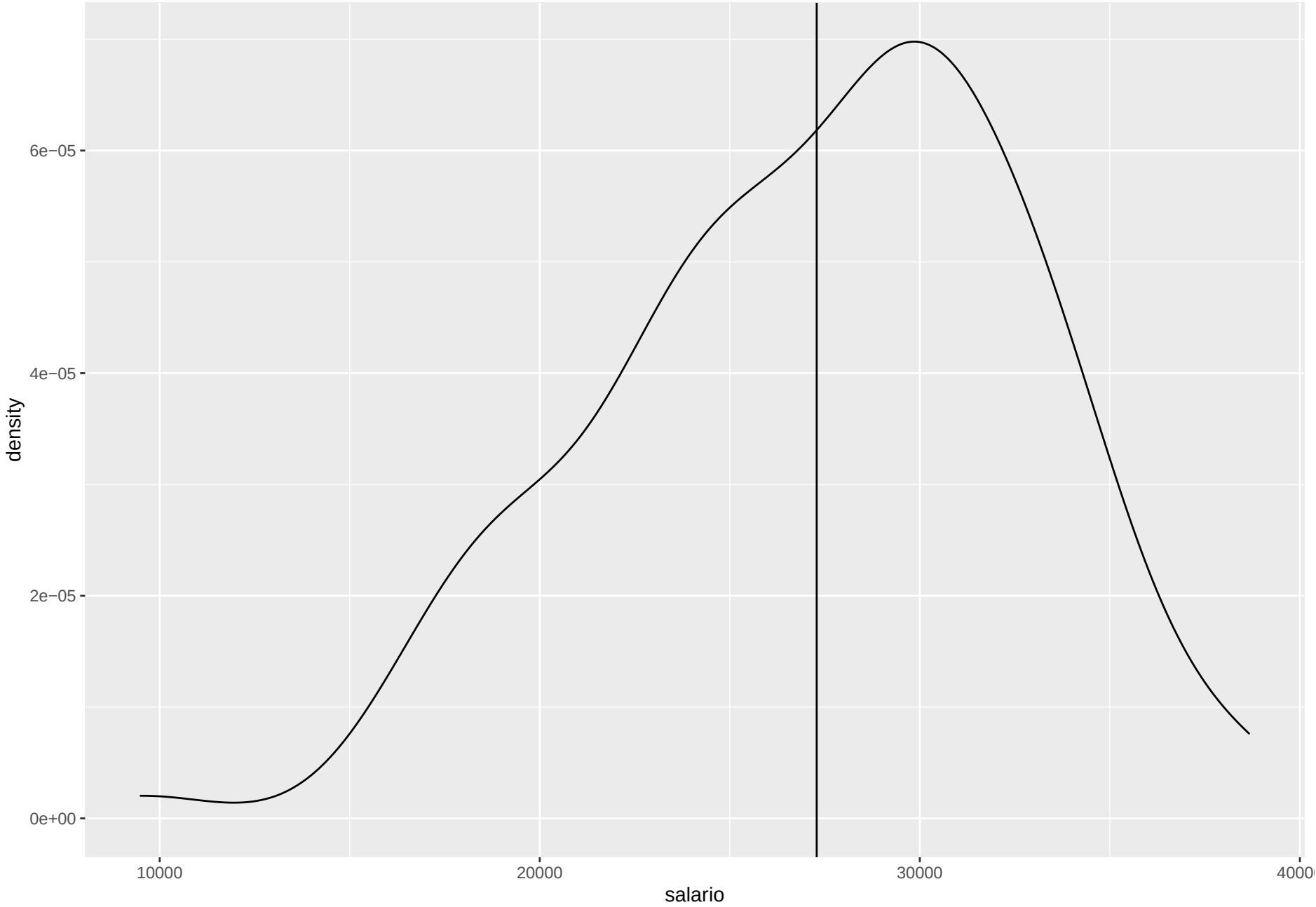
```
set.seed(2022)

n <- 100
mean_h <- 30000
mean_m <- 24000
sd_h <- 3000
sd_m <- 5000



salarios <- tibble(
  id=seq(n),
  sexo= factor(rep(c('m','h'), each= n/2)),
  salario= round(c(rnorm(n/2, mean_m, sd_m),
                   rnorm(n/2, mean_h, sd_h)))
)
```

	id	sexo	salario
1	1	m	28501.00
2	2	m	18133.00
3	3	m	19513.00
4	4	m	16777.00
5	5	m	22345.00
6	6	m	9497.00

Densidad

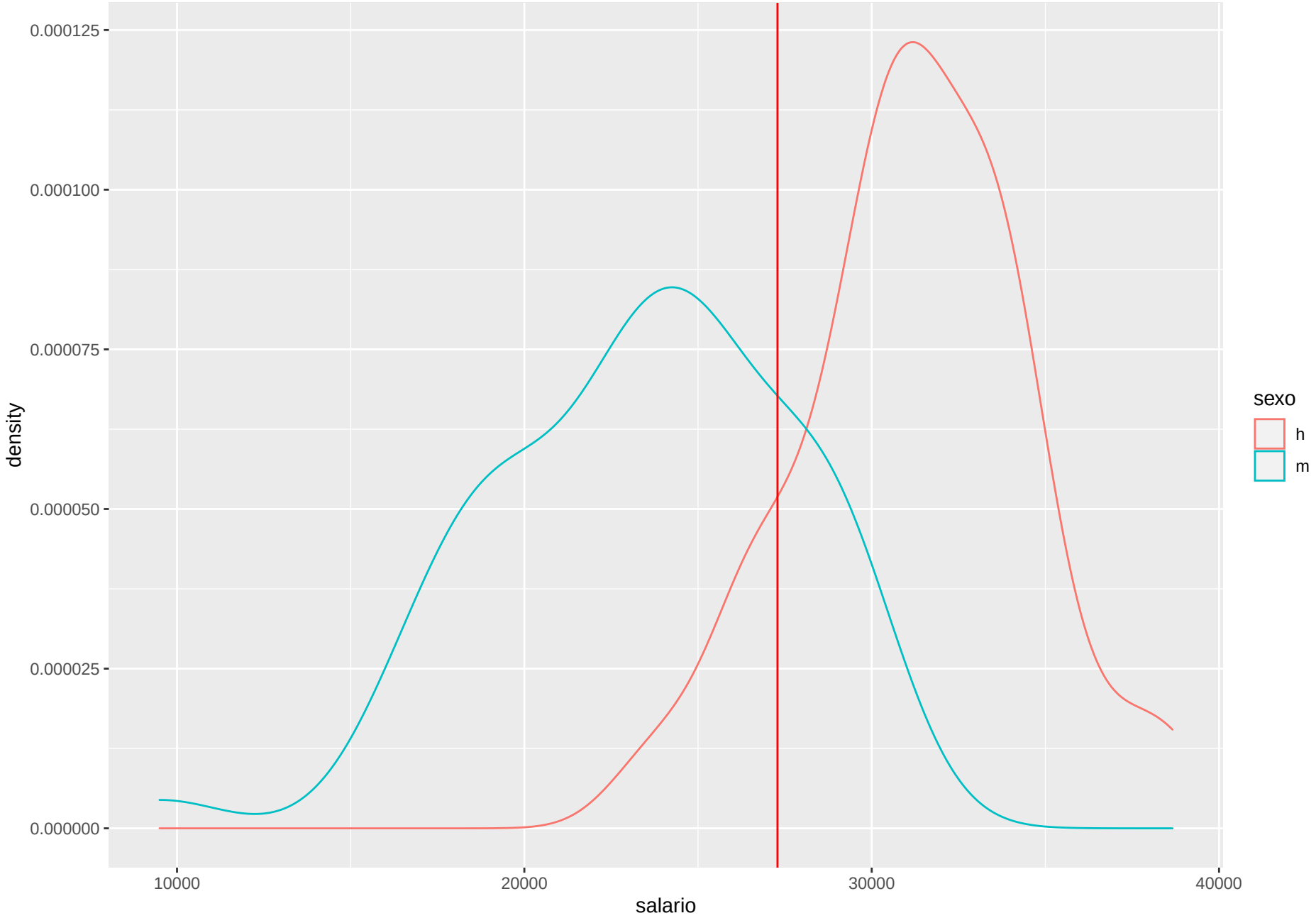
- Hacemos la gráfica de densidad con geom_density()
- Como es densidad, solo necesitamos especificar una variable
- Añadamos una línea vértical que cruce en la media
- las líneas horizontales o verticales son una geometría en sí
  - geomvline() para líneas verticales
  - geomhline() para líneas horizontales
- Si queremos que el corte de la línea dependa de los datos, tenemos que agregar la media en la estética de geom_vline()

```
(density <- ggplot(salarios,
                   aes(salario))+
    geom_density()+
    geom_vline(aes(xintercept=mean(salario))))
```


6e−05
4e−05
2e−05
0e+00
density
10000
20000
30000
salario

Densidad por grupo

```
(density_group <- ggplot(salarios,
                         aes(salario, color= sexo))+
    geom_density()+
    geom_vline(aes(xintercept=mean(salario)),
                   color='red'))
```

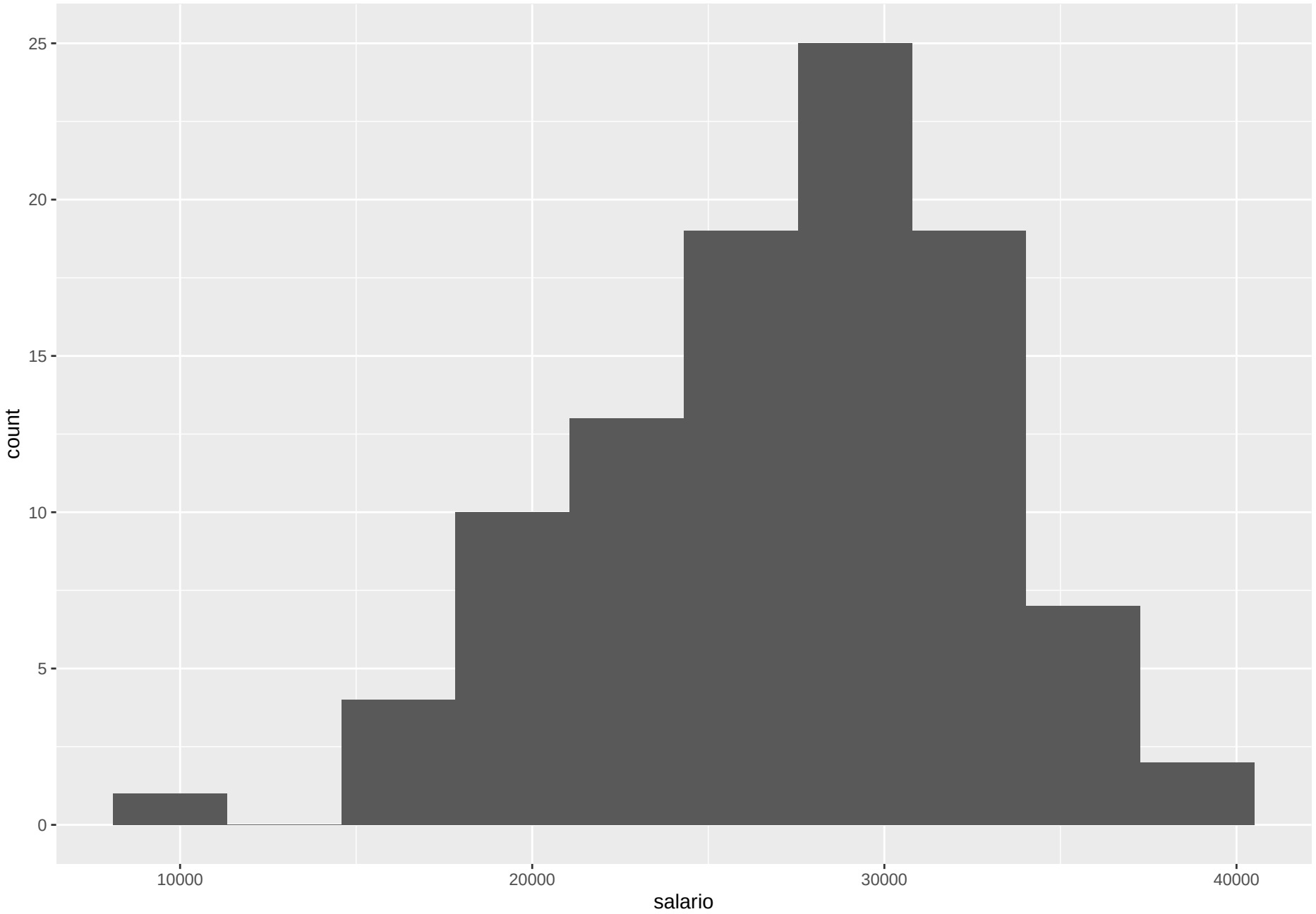

0.000125
0.000100
0.000075
0.000050
0.000025
0.000000
density
10000
20000
30000
40000
salario
sexo
h
m

Histograma

- Evidentemente, el histograma es muy similar a la densidad
- Podemos, además, especificar las marcas de clase
  - bins= # ajusta el número de marcas de clase a las especificadas
  - binwidth= # crea las marcas de la longitud deseada

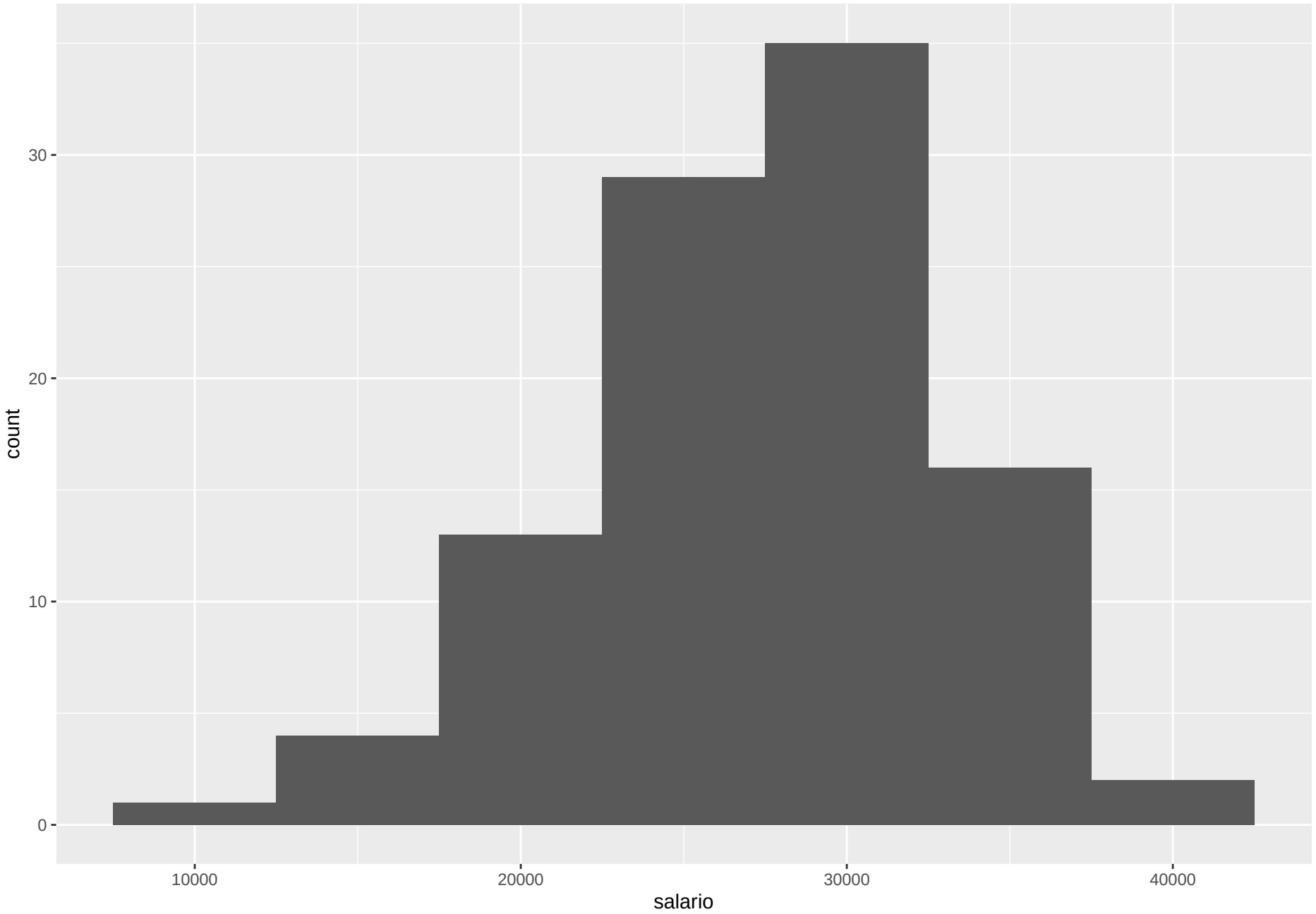
- Repliquemos lo mismo de la densidad, pero con el histograma
- Probemos primero con el número de marcas de clase

```
(histogram_b <- ggplot(salarios,
                       aes(salario))+
    geom_histogram(bins=10))
```


25
20
15
10
5
0
count
10000
20000
30000
40000
salario

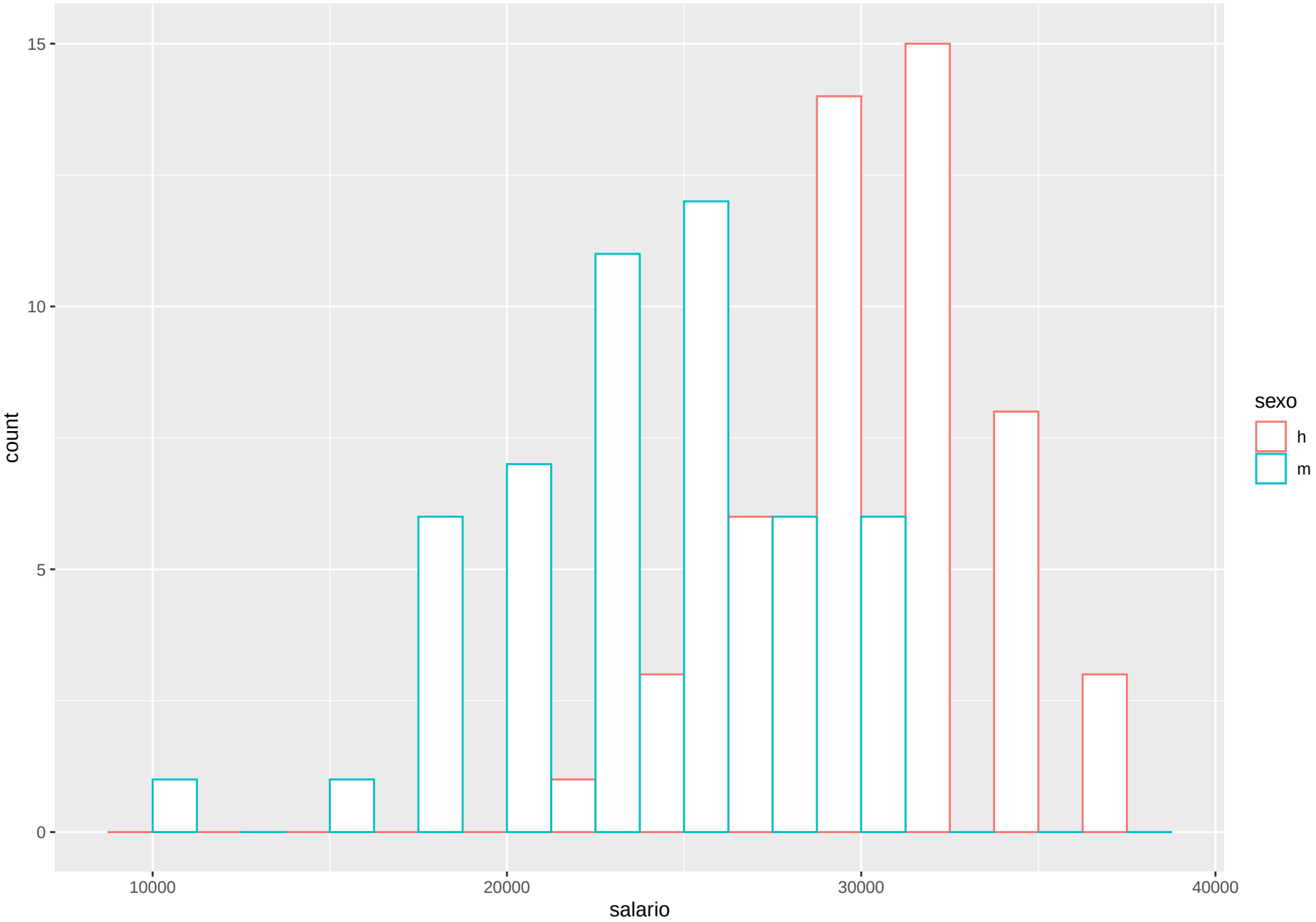
- Ahora con la longitud de clase

```
(histogram_bwidth <- ggplot(salarios,
                            aes(salario))+
    geom_histogram(binwidth = 5000))
```


30
20
10
0
count
10000
20000
30000
40000
salario

- Así como con la densidad, podemos distinguir por color

```
(histogram_sexo <-
    ggplot(salarios,
           aes(salario, color= sexo))+
    geom_histogram(binwidth = 2500,
                   position= 'dodge',
                   fill='white'))
```


15
10
5
0
count
10000
20000
30000
40000
salario
sexo
h
m

Section 10

ggsave()

ggsave()

- Los gráficos de ggplot() pueden guardarse como cualquier objeto
- Podemos guardar nuestra gráfica generada como un archivo de formato de nuestra preferencia con ggsave()
- Ejecutarlo una y otra vez no genera un nuevo archivo, sino que lo sobreescribe
- Podemos modificar tamaño o escala

```
ggsave(
  filename, # nombre para guardar
  plot = last_plot(), # grafica que va a guardar (ultima pred
  device = NULL, # formato
  path = NULL, # donde lo va a guardar
)
```