Exploración de Datos con R

Rubén Sierra Serrano

2024-04-12

Índice

Exploración de datos categóricos

En el siguiente documento se va a trabajar con el DataFrame `comics`.

La función `prop.table` se utiliza para crear tablas de frecuencia relativa a partir de tablas de frecuencia absoluta, la estructura de la función se muestra a continuación.

prop.table(x, margin=NULL) x: tabla de frecuencia. margin: valor de 1 si se desean proporciones por filas, 2 si se desean por columnas, NULL si se desean frecuencias globales.

```
options(scipen = 999, digits = 3)
tabla_cnt <- table(comics$id, comics$align)
tabla_cnt
```

```
##
##            Bad Good Neutral Reformed Criminals
##   No Dual  474  647     390                   0
##   Public  2172 2930     965                   1
##   Secret  4493 2475     959                   1
##   Unknown    7    0       2                   0
```

```
prop.table(tabla_cnt)
```

```
##
##                 Bad      Good   Neutral Reformed Criminals
##   No Dual 0.0305491 0.0416989 0.0251353          0.0000000
##   Public  0.1399845 0.1888373 0.0621939          0.0000644
##   Secret  0.2895721 0.1595128 0.0618072          0.0000644
##   Unknown 0.0004511 0.0000000 0.0001289          0.0000000
```

```
prop.table(tabla_cnt, 1)
```

```
##
##                Bad     Good  Neutral Reformed Criminals
##   No Dual 0.313700 0.428193 0.258107           0.000000
##   Public  0.357943 0.482861 0.159031           0.000165
##   Secret  0.566726 0.312185 0.120964           0.000126
##   Unknown 0.777778 0.000000 0.222222           0.000000
```

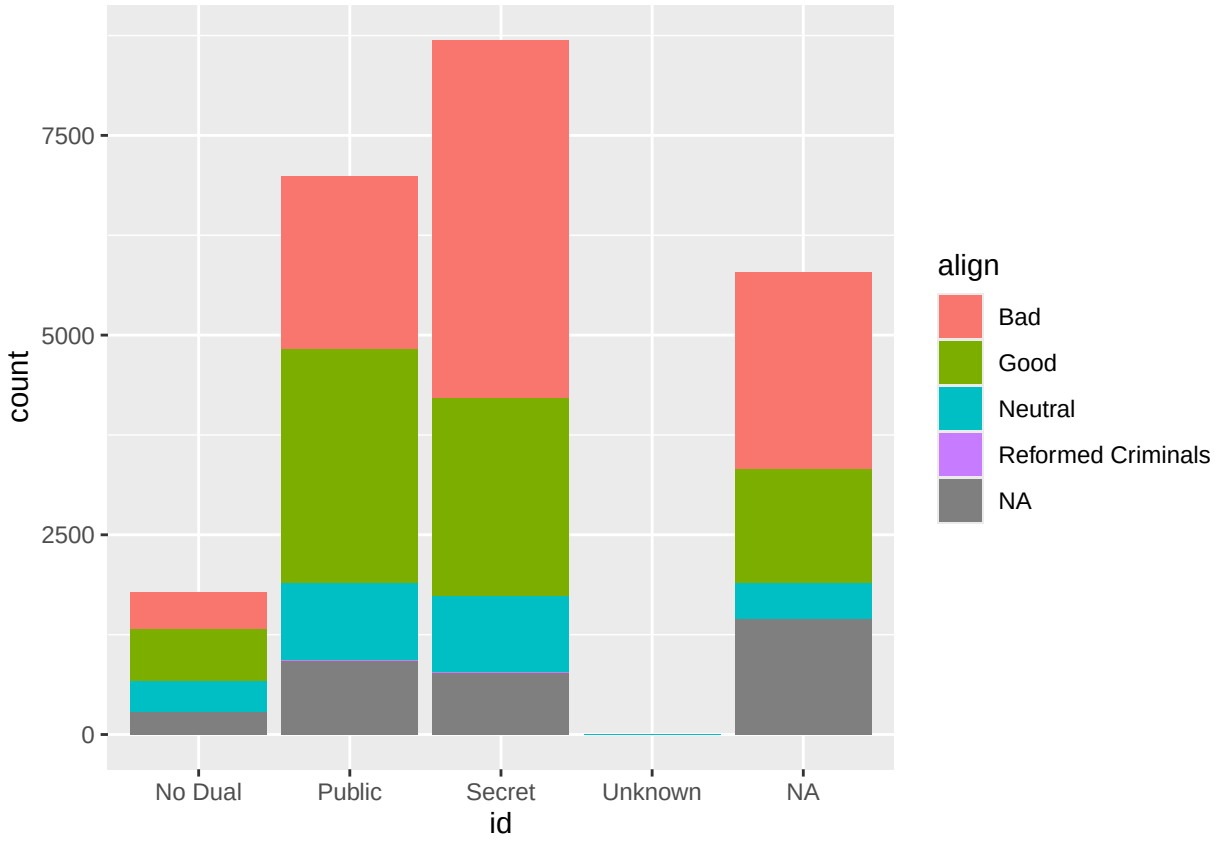
```
prop.table(tabla_cnt, 2)
```

```
##
##                Bad     Good  Neutral Reformed Criminals
##   No Dual 0.066331 0.106907 0.168394           0.000000
##   Public  0.303946 0.484137 0.416667           0.500000
##   Secret  0.628743 0.408956 0.414076           0.500000
##   Unknown 0.000980 0.000000 0.000864           0.000000
```

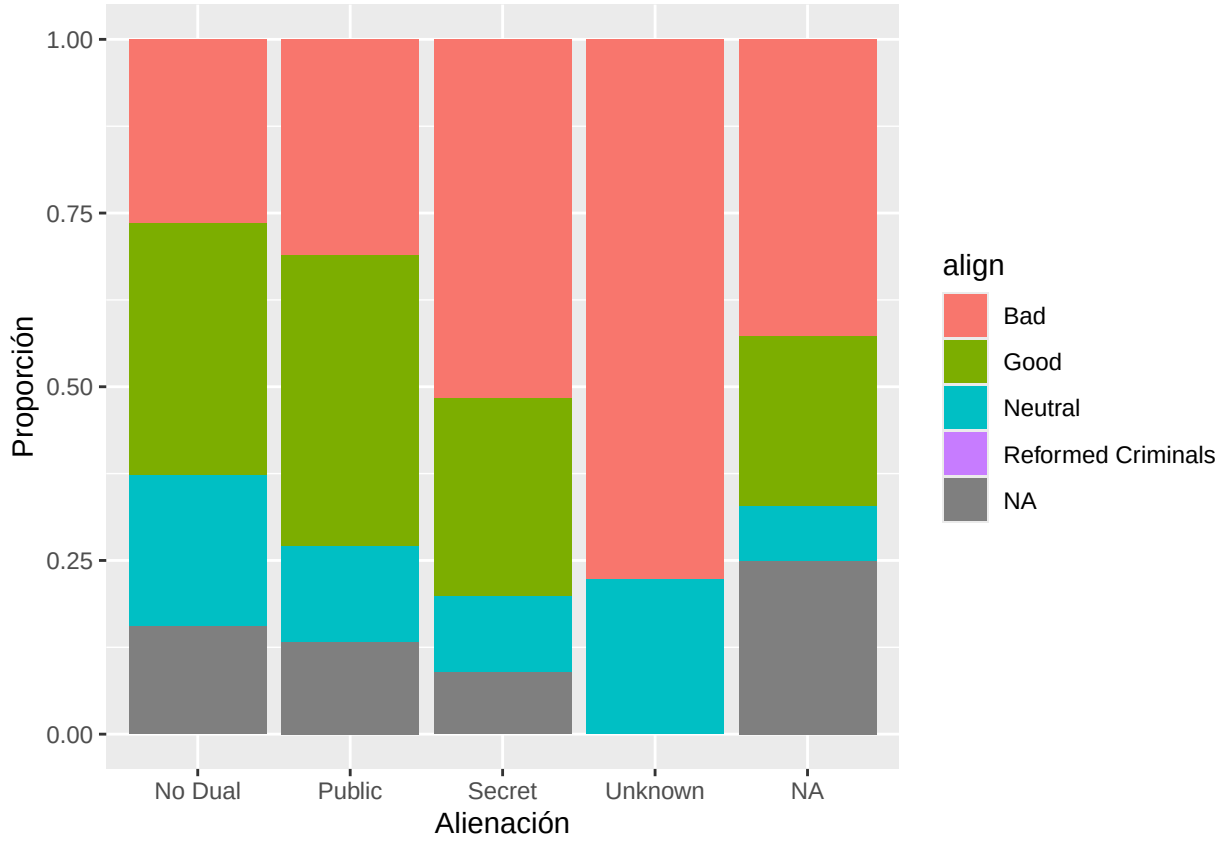
```
prop.table(tabla_cnt, NULL)
```

```
##
##                 Bad      Good   Neutral Reformed Criminals
##   No Dual 0.0305491 0.0416989 0.0251353          0.0000000
##   Public  0.1399845 0.1888373 0.0621939          0.0000644
##   Secret  0.2895721 0.1595128 0.0618072          0.0000644
##   Unknown 0.0004511 0.0000000 0.0001289          0.0000000
```

```
ggplot(comics, aes(x = id, fill = align)) +
  geom_bar()
```




```
ggplot(comics, aes(x = id, fill = align)) +
  geom_bar(position = "fill") +
  ylab("Proporción") +
  xlab("Alienación")
```



Eliminar categorías

Para eliminar categorías que no nos resultan útiles a la hora de explorar los datos, ya sea porque no las estamos estudiando, contienen outliers, tienen muchos NaNs, etc., podemos emplear la función **droplevels()** y el paquete `dplyr` de la siguiente forma:

```
comics_filtered <- comics %>%
  filter(align != "Reformed Criminals") %>%
  filter(align != "Unknown") %>%
  filter(align != "NA") %>%
  droplevels()
```

Distribución de una variable

Para calcular la tabla de conteos de una variable podemos emplear la función **table** e introducir como parámetro un vector de la siguiente forma:

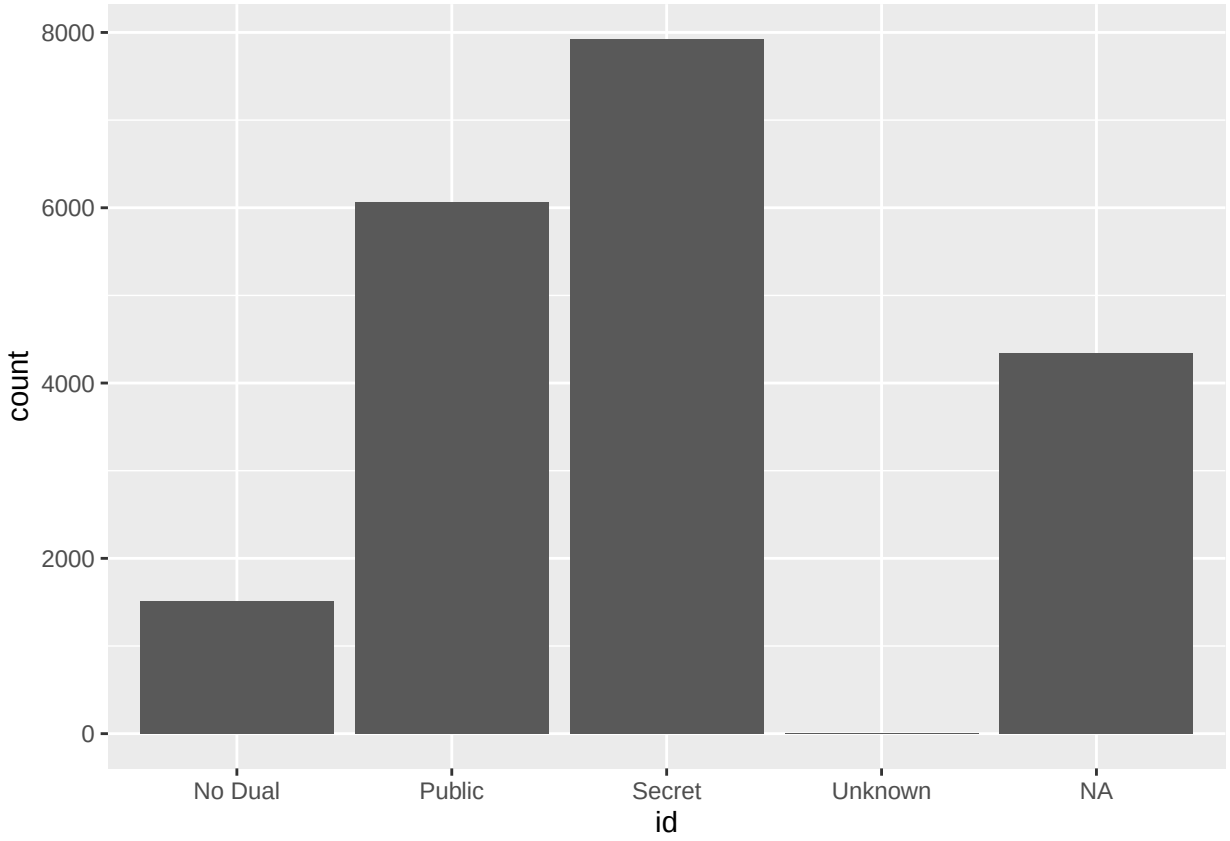
```
table(comics_filtered$id)
```

```
##
## No Dual  Public  Secret Unknown
##    1511    6067    7927       9
```

Esto se conoce como la distribución marginal de la variable `id`.

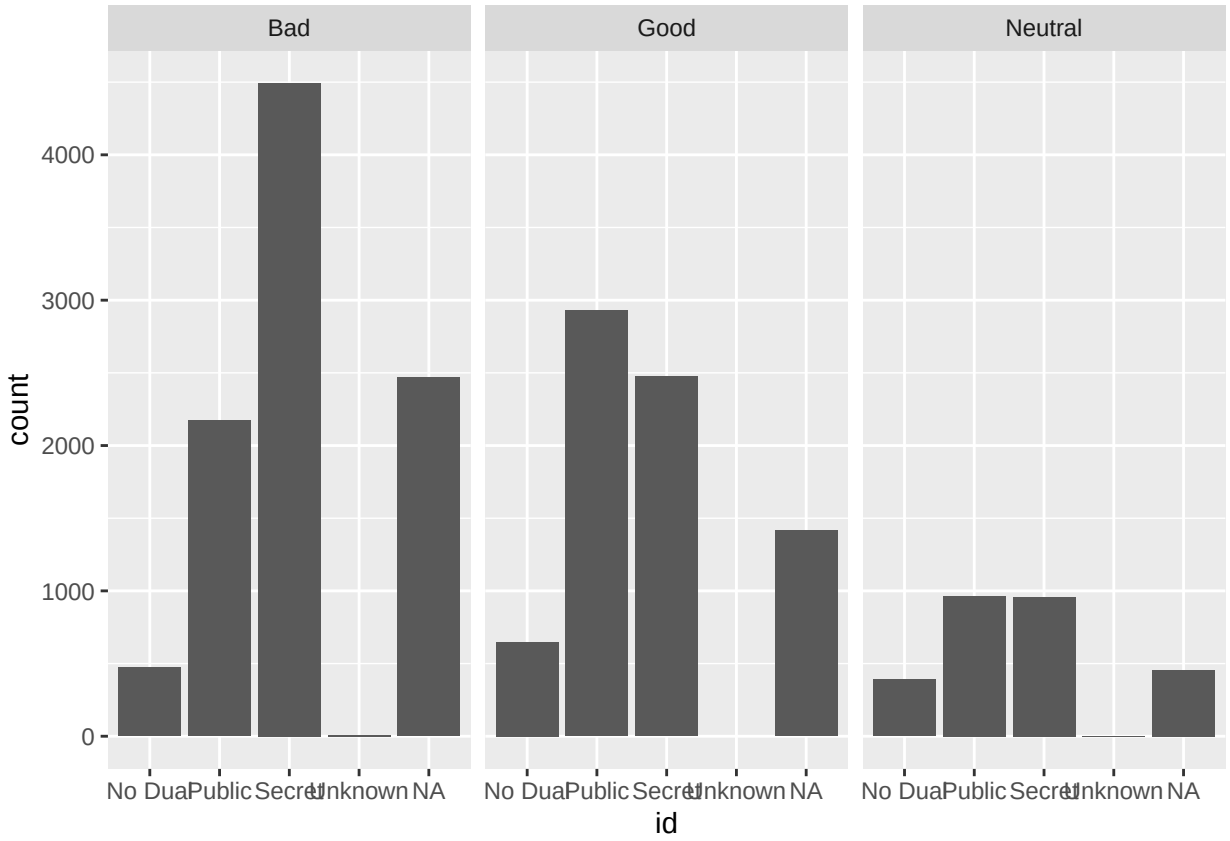
Resulta sencillo crear el gráfico de barras de dicha distribución marginal:

```
ggplot(comics_filtered, aes(x = id)) +
  geom_bar()
```



Otra manera útil de obtener la distribución de una variable es condicionarla a un valor particular de otra variable (faceting: divide los datos en subconjuntos en función de los niveles de una variable categórica). Por ejemplo, podríamos estar interesados en la variable `id` de todos los personajes de la categoría `Neutral` de la variable `align`; para ello:

```
ggplot(comics_filtered, aes(x = id)) +
  geom_bar() +
  facet_wrap(~align)
```



Exploración de datos numéricos

Para realizar la exploración de datos numéricos emplearemos el dataset `cars`.

```
str(cars)
```

```
## 'data.frame':    428 obs. of  19 variables:
##  $ name       : chr  "Chevrolet Aveo 4dr" "Chevrolet Aveo LS 4dr hatch" "Chevrolet Cavalier 2dr" "Ch
##  $ sports_car : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ suv        : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ wagon      : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ minivan    : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ pickup     : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ all_wheel  : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ rear_wheel : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ msrp       : int  11690 12585 14610 14810 16385 13670 15040 13270 13730 15460 ...
##  $ dealer_cost: int  10965 11802 13697 13884 15357 12849 14086 12482 12906 14496 ...
##  $ eng_size   : num  1.6 1.6 2.2 2.2 2.2 2 2 2 2 2 ...
##  $ ncyl       : int  4 4 4 4 4 4 4 4 4 4 ...
##  $ horsepwr   : int  103 103 140 140 140 132 132 130 110 130 ...
##  $ city_mpg   : int  28 28 26 26 26 29 29 26 27 26 ...
##  $ hwy_mpg    : int  34 34 37 37 37 36 36 33 36 33 ...
##  $ weight     : int  2370 2348 2617 2676 2617 2581 2626 2612 2606 2606 ...
##  $ wheel_base : int  98 98 104 104 104 105 105 103 103 103 ...
```

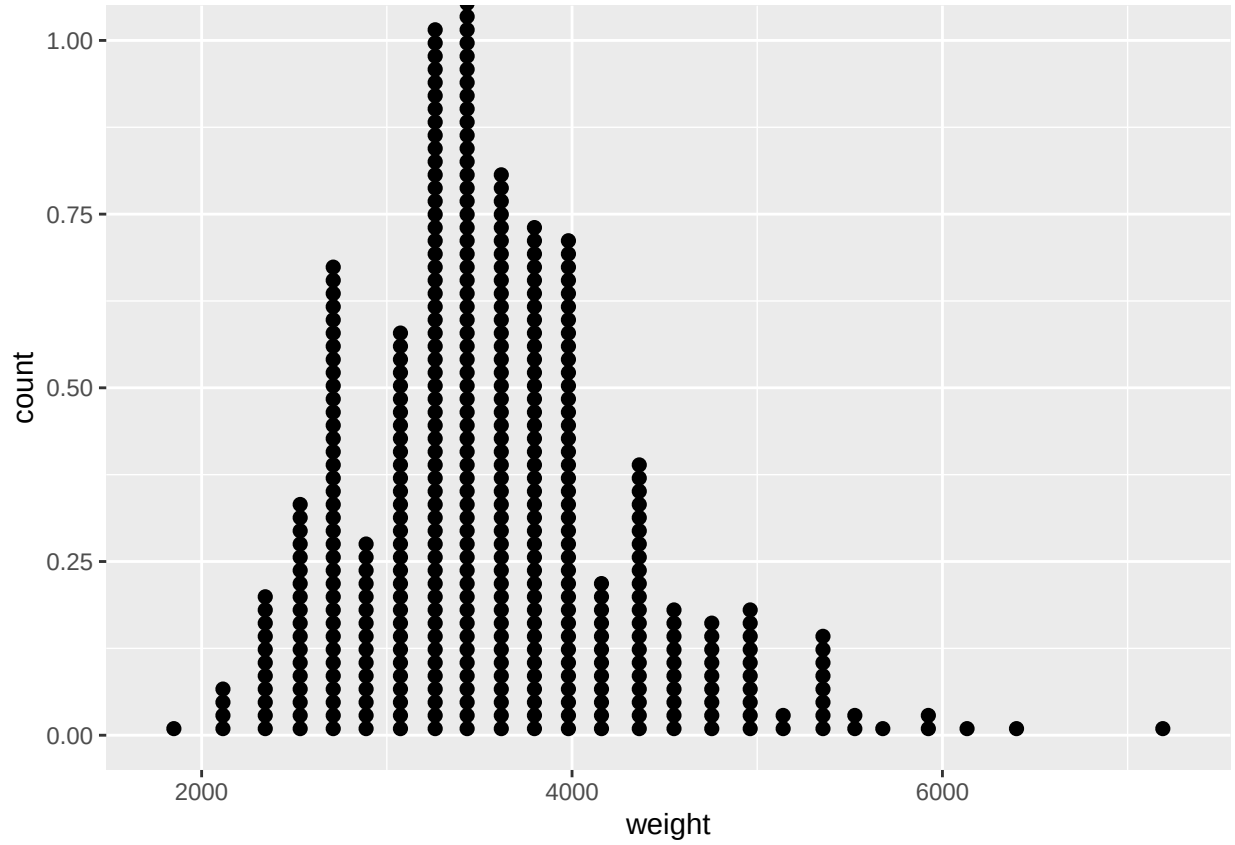
```
##  $ length     : int  167 153 183 183 183 174 174 168 168 168 ...
##  $ width      : int  66 66 69 68 69 67 67 67 67 67 ...
```

Una primera aproximación útil a un dato numérico es mediante un diagrama de puntos en el que cada caso es representado por un punto:

```
ggplot(cars, aes(x = weight)) +
  geom_dotplot(dotsize = 0.4)
```

```
## Bin width defaults to 1/30 of the range of the data. Pick better value with
## `binwidth`.
```

```
## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`stat_bindot()`).
```

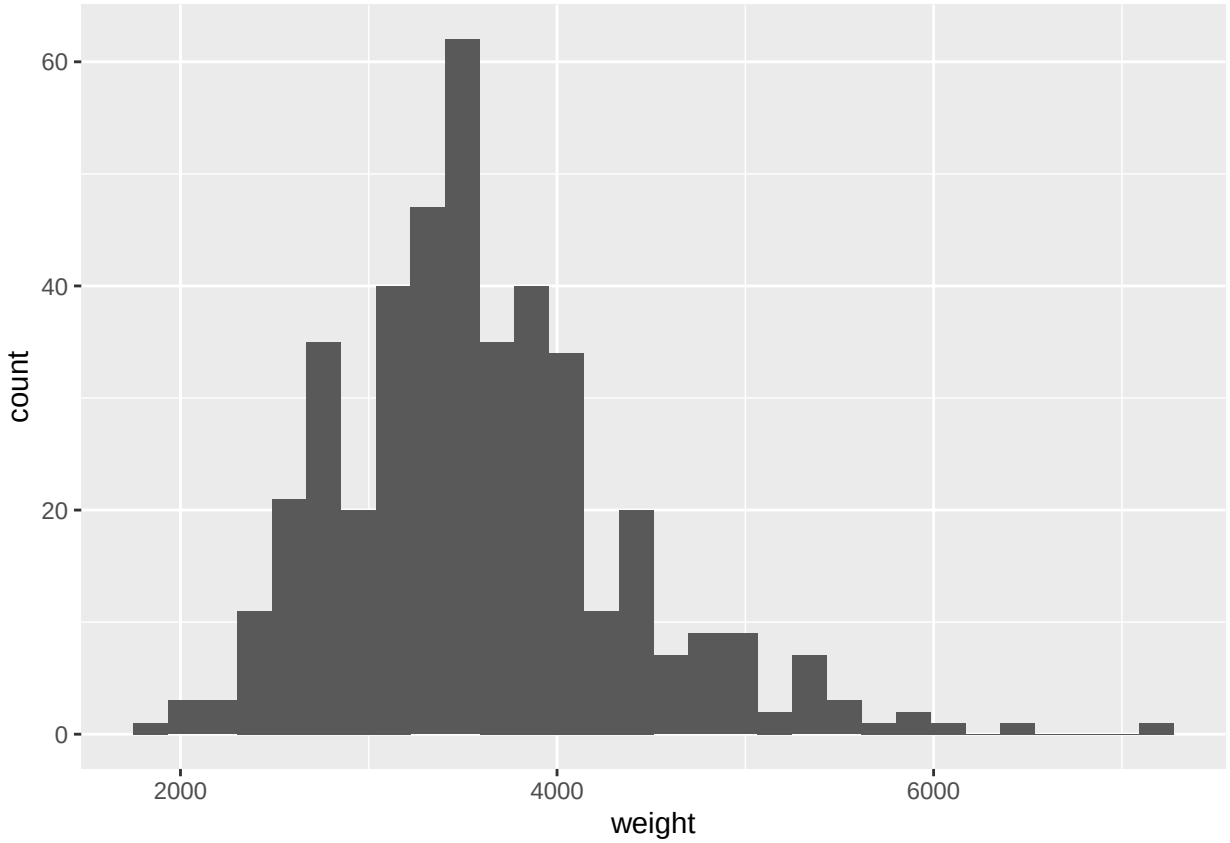


Este es un tipo de gráfico en el que no hay pérdida de información; sin embargo, para datasets de gran tamaño, los puntos se solapan y pueden dificultar una lectura efectiva del gráfico. En estos casos, los histogramas pueden ser de gran utilidad, ya que agrupan los datos en barras de las cuales obtenemos su frecuencia (el número de barras puede ser modificado por el parámetro `bins`):

```
ggplot(cars, aes(x = weight)) +
  geom_histogram()
```

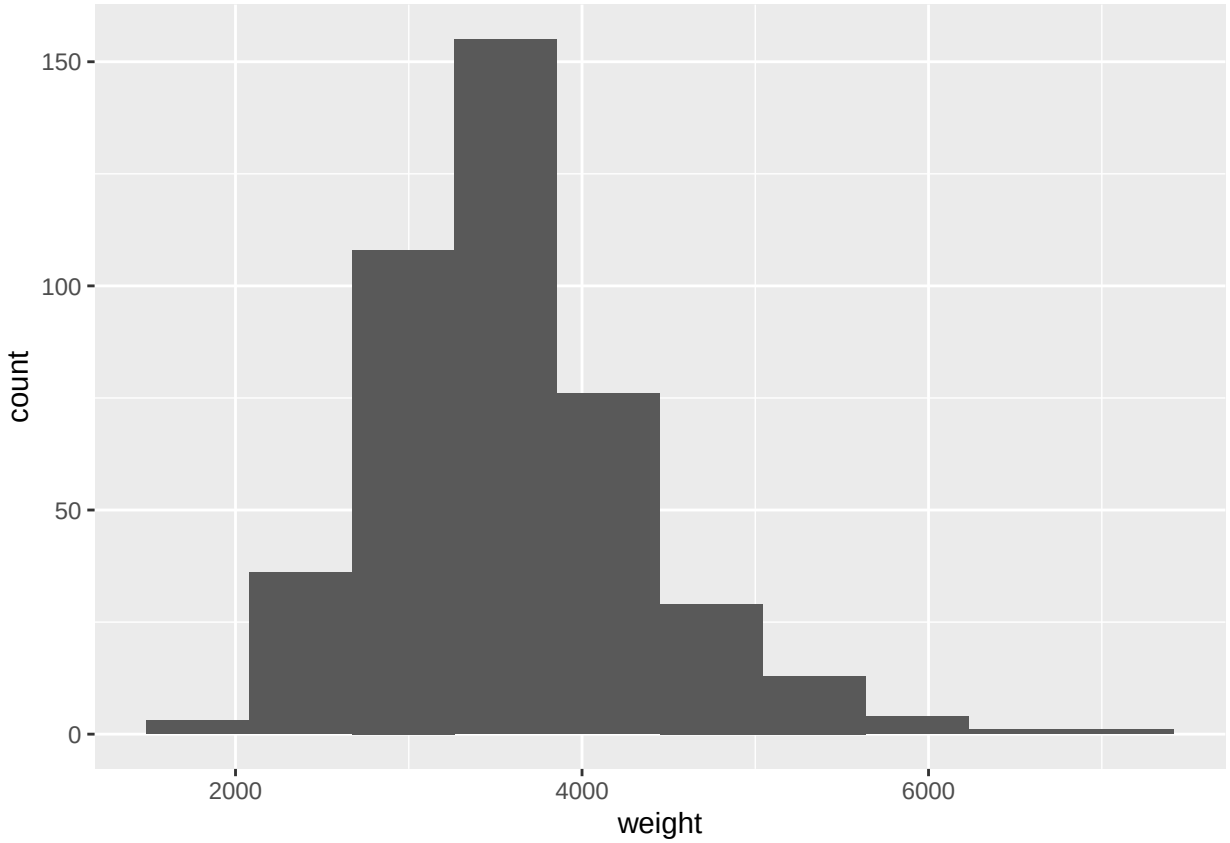
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 2 rows containing non-finite outside the scale range
## (‘stat_bin()‘).
```




```
ggplot(cars, aes(x = weight)) +
  geom_histogram(bins = 10)
```

```
## Warning: Removed 2 rows containing non-finite outside the scale range
## (‘stat_bin()‘).
```



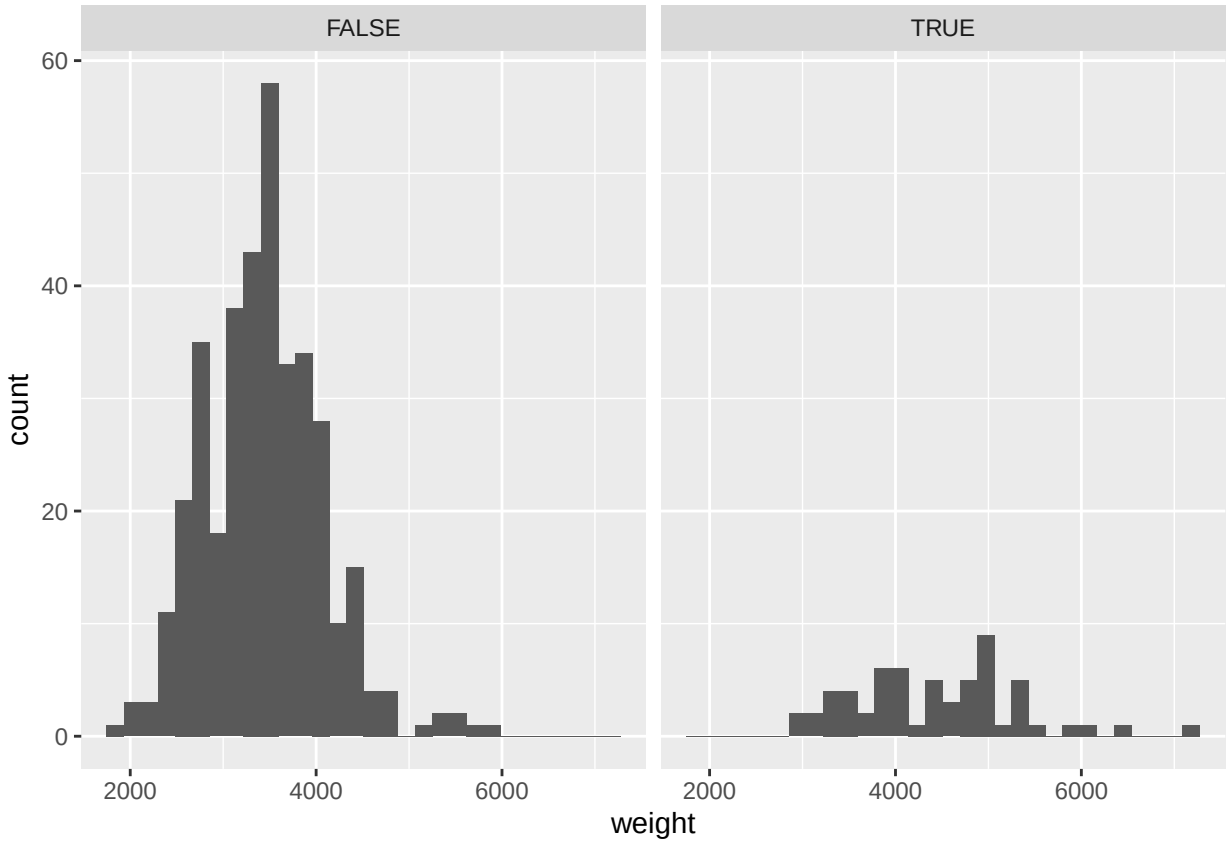
En los histogramas si que hay cierta pérdida de información pero permiten contemplar de manera más precisa la distribución de la variable estudiada.

Se pueden realizar también histogramas de una categoría en concreto gracias a las facetas:

```
ggplot(cars, aes(x = weight)) +
  geom_histogram() +
  facet_wrap(~suv)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

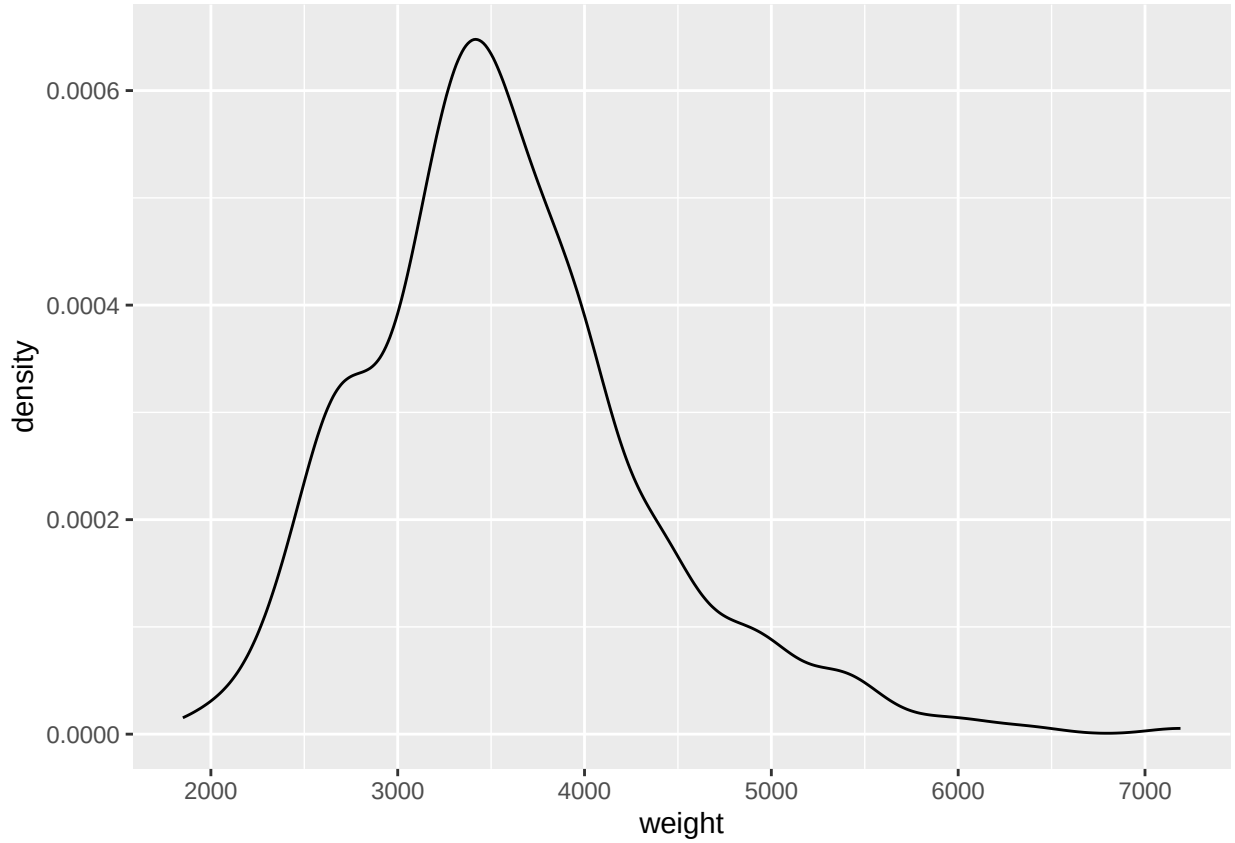
```
## Warning: Removed 2 rows containing non-finite outside the scale range
## (`stat_bin()`).
```



Una gráfica muy similar al histograma es la función de densidad que, en contraposición a la naturaleza discreta del histograma, es continua al representar la distribución de la variable empleando una línea suave:

```
ggplot(cars, aes(x = weight)) +
  geom_density()
```

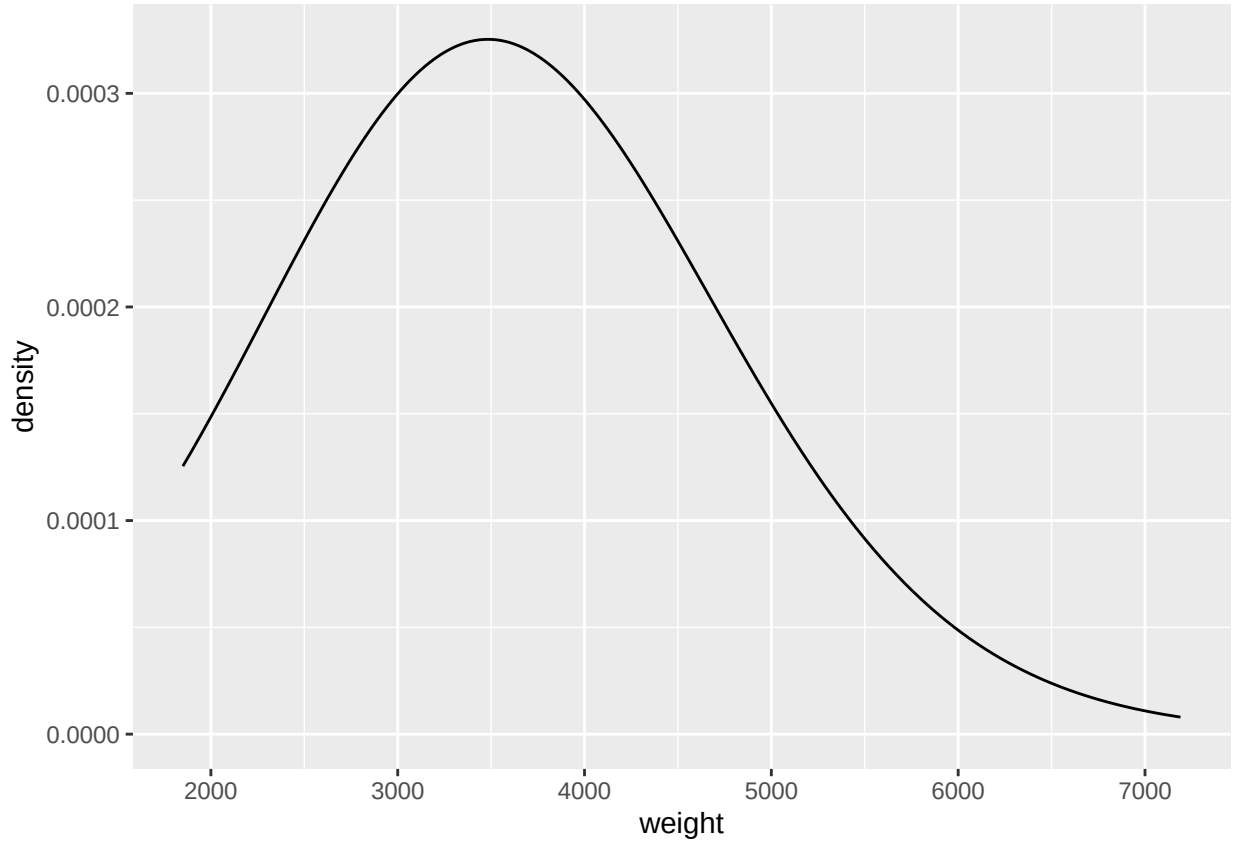
```
## Warning: Removed 2 rows containing non-finite outside the scale range
## (‘stat_density()‘).
```



Para modificar la estimación de densidad del kernel (aplicación de suavizado para la estimación de la densidad), podemos modificar el argumento `bw`(bandwidth) de la función:

```
ggplot(cars, aes(x = weight)) +
  geom_density(bw = 1000)
```

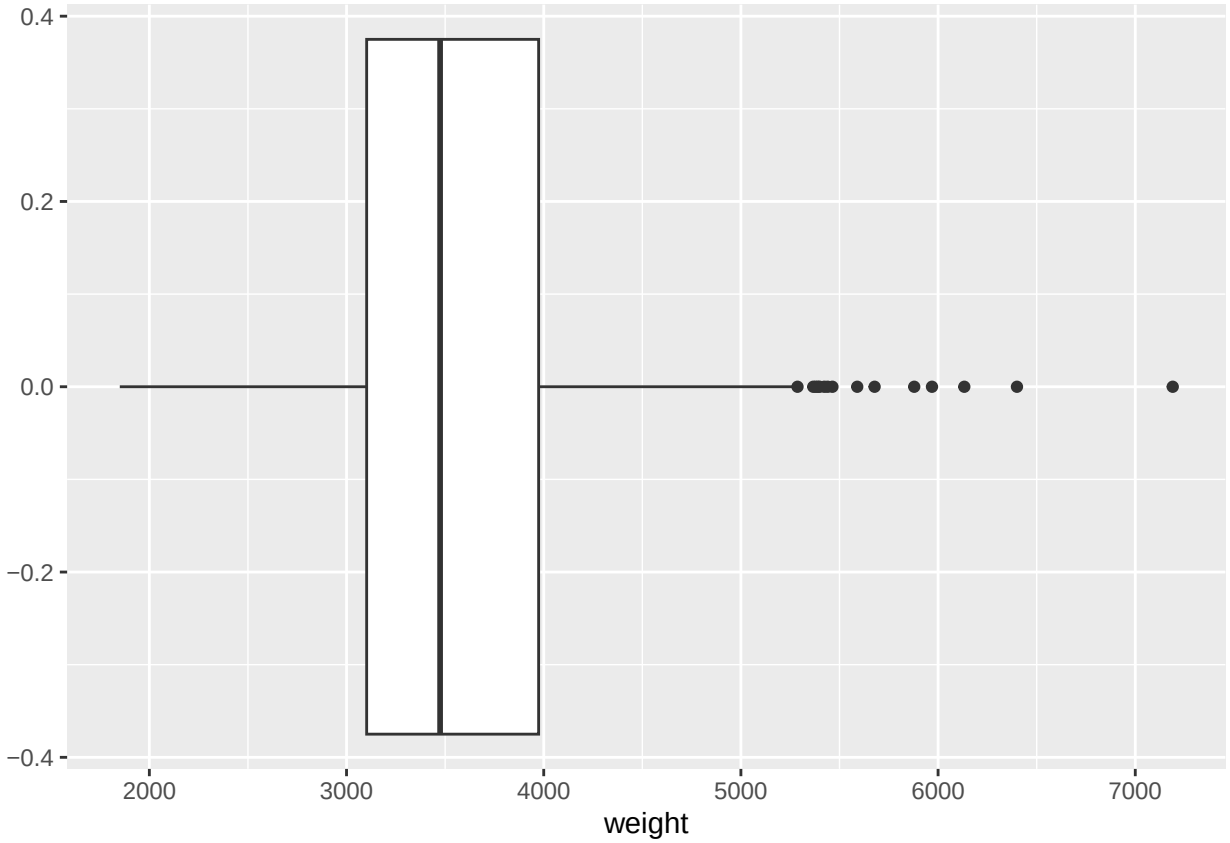
```
## Warning: Removed 2 rows containing non-finite outside the scale range
## (‘stat_density()‘).
```



Otro gráfico interesante que muestra la naturaleza de la variable es el gráfico de caja y bigotes en el que se muestra la dispersión y la simetría, así como los cuartiles, la media, la mediana y los outliers:

```
ggplot(cars, aes(x = weight)) +
  geom_boxplot()
```

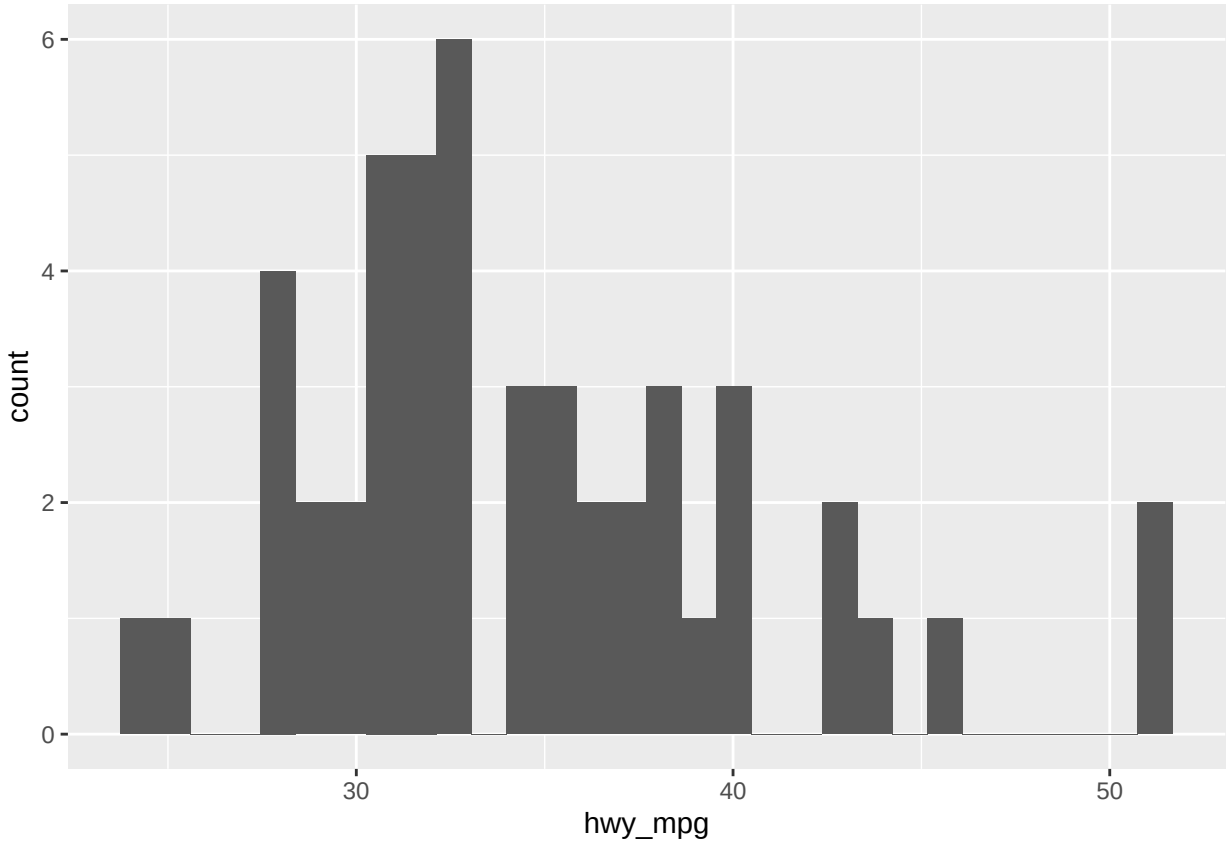
```
## Warning: Removed 2 rows containing non-finite outside the scale range
## (‘stat_boxplot()‘).
```



Podemos filtrar valores de una variable a la hora de graficar:

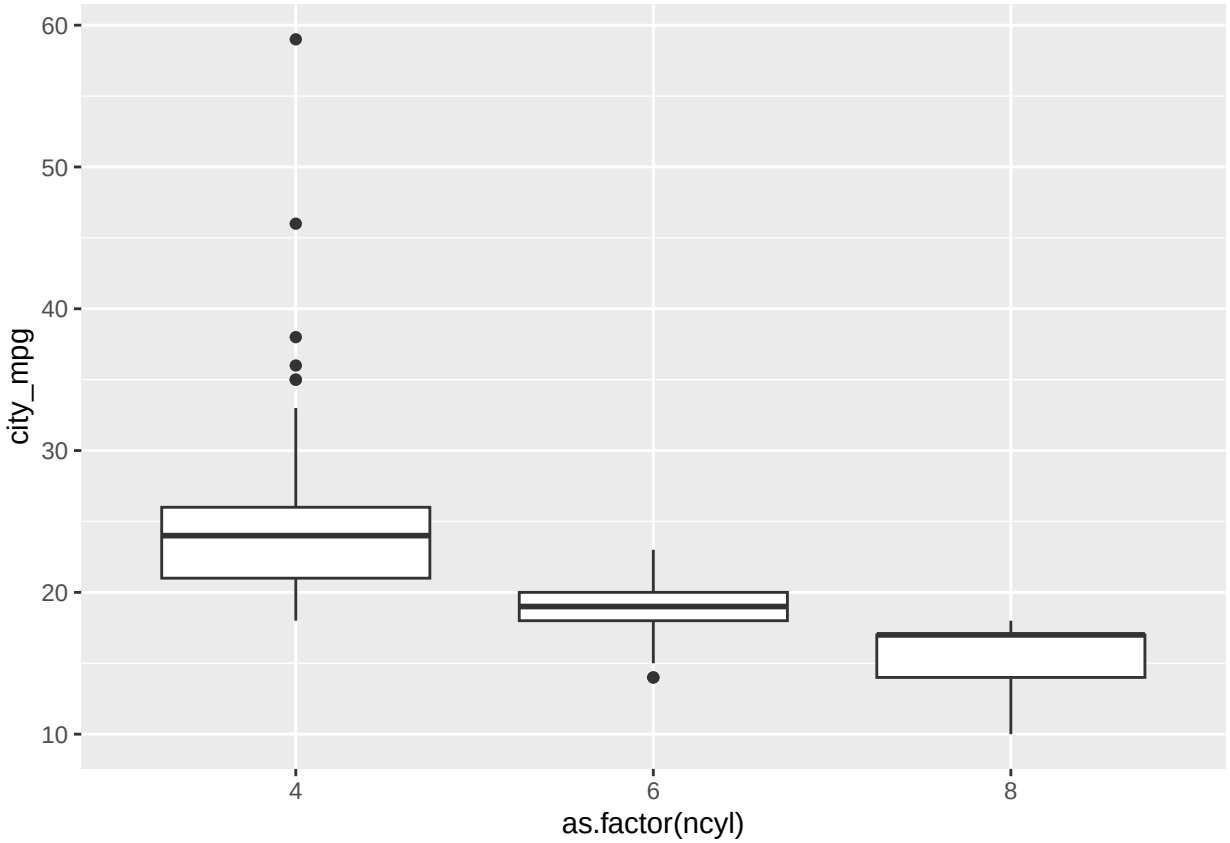
```
cars %>%
  filter(eng_size < 2.0) %>%
  ggplot(aes(x = hwy_mpg)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



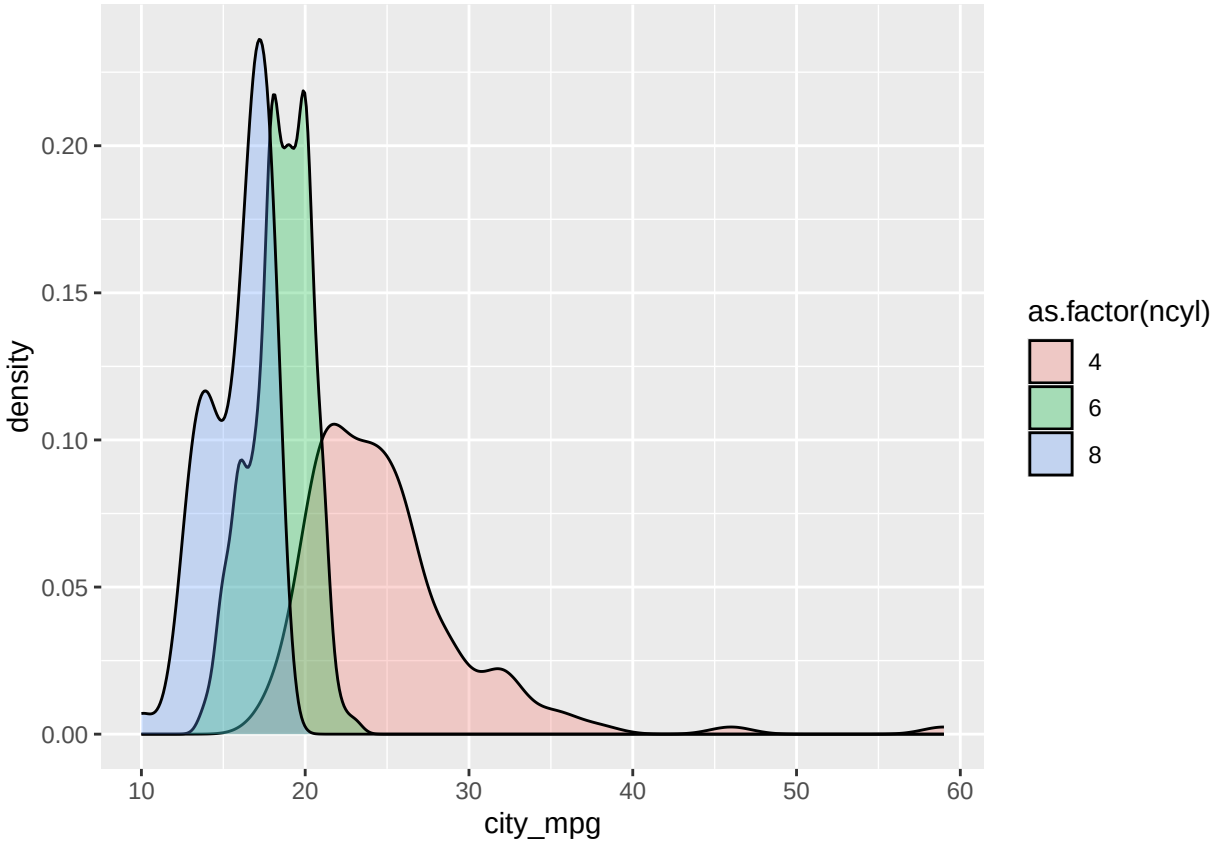

```
common_cyl <- filter(cars, ncyl %in% c(4, 6, 8))

ggplot(common_cyl, aes(x = as.factor(ncyl), y = city_mpg)) +
  geom_boxplot()
```

```
## Warning: Removed 11 rows containing non-finite outside the scale range
## (`stat_boxplot()`).
```




```
ggplot(common_cyl, aes(x = city_mpg, fill = as.factor(ncyl))) +
  geom_density(alpha = .3)
```

```
## Warning: Removed 11 rows containing non-finite outside the scale range
## (`stat_density()`).
```



Visualización en dimensiones más altas