# Tarea1

Gonzalo Ruiz Espinar

9/15/2021

## Ejercicio 0

Empezaremos cargamos la libreria *tidyverse*.

```
library(tidyverse)
```

Usando la función sample para crear un dado honesto con 100 números del 1 al 6.

```
dado_honesto=sample(1:6, size = 100, replace = TRUE)
dado_honesto
```

```
##   [1] 6 3 3 4 5 2 6 4 4 2 1 2 2 4 4 1 5 3 1 2 1 1 6 6 3 4 2 3 3 5 2 1 2 1
2 6 5
##  [38] 1 1 2 1 1 2 2 1 4 3 5 2 5 6 1 6 6 2 3 3 4 2 6 4 3 4 1 6 5 3 2 4 3 2
5 5 5
##  [75] 6 5 6 4 3 5 5 1 5 4 1 5 6 6 1 6 2 6 6 6 3 1 4 5 2 3
```

Dado que se trata de una variable discreta, realizamos una tabla de frecuencia absoluta con el `tidyverse`:

```
table(dado_honesto)
```

```
## dado_honesto
##  1  2  3  4  5  6
## 18 19 15 14 16 18
```

Y una tabla de frecuencias absolutas con el R básico:

```
dado <-
  data.frame(A = 1:100, dado_A = dado_honesto)

dado %>% count(Dado=dado$dado_A)
```

```
##   Dado  n
## 1    1 18
## 2    2 19
## 3    3 15
## 4    4 14
## 5    5 16
## 6    6 18
```

Una tabla de frecuencias relativas con R básico:

```
signif(prop.table(table(dado$dado_A)), 2)
```

```
##
##    1    2    3    4    5    6
## 0.18 0.19 0.15 0.14 0.16 0.18
```

y una tabla de frecuencias relativas con `tidyverse`:

```
dado %>%
  count(dado_A) %>%
  mutate(dado_A, relFreq = prop.table(n), n=NULL)
```

```
##   dado_A relFreq
## 1      1    0.18
## 2      2    0.19
## 3      3    0.15
## 4      4    0.14
## 5      5    0.16
## 6      6    0.18
```

A continuación creamos un dado cargado de manera que la probabilidad de que el número elegido valga 6 sea el doble que la probabilidad de elegir cualquiera de los cinco números restantes:

```
pesos=c(1/7, 1/7, 1/7, 1/7, 1/7, 2/7)
dado_cargado=sample(1:6, size = 100, replace = TRUE, prob=pesos)
dado_cargado
```

```
##   [1] 2 4 2 5 4 5 4 1 3 2 4 3 3 5 4 3 1 2 3 5 4 4 4 6 6 6 6 5 6 6 1 3 5 6
1 1 6
##  [38] 3 3 1 1 6 6 2 2 6 6 6 6 6 6 1 6 2 6 1 2 3 5 4 3 6 6 2 3 1 2 1 2 4 4
4 5 6
##  [75] 2 4 6 1 1 2 5 6 6 1 2 6 1 4 2 6 2 3 1 6 5 5 1 3 6 2
```

Usamos las funciones `rep()` y `seq()` para crear los siguientes vectores:

```
rep(4:1,each=4)
```

```
##  [1] 4 4 4 4 3 3 3 3 2 2 2 2 1 1 1 1
```

```
rep(1:5,times=seq(1,5))
```

```
##  [1] 1 2 2 3 3 3 4 4 4 4 5 5 5 5 5
```

```
rep(1:4,4)
```

```
##  [1] 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4
```

Utilizamos la tabla `mpg` para seleccionar las columnas cuyos nombres empiezan por c, y que las filas en las que la variable class toma el valor pickup.

```
mpg %>% select (starts_with("c")) %>% filter(class == "pickup") -> mpg2
head(mpg2)
```

```
## # A tibble: 6 × 3
##     cyl   cty class
##   <int> <int> <chr>
## 1     6    15 pickup
## 2     6    14 pickup
## 3     6    13 pickup
## 4     6    14 pickup
## 5     8    14 pickup
## 6     8    14 pickup
```

Cargamos la tabla census:

```
library(haven)
census <- read_dta("data/census.dta")
head(census)
```

```
## # A tibble: 6 × 12
##   state        region    pop poplt5 pop5_17 pop18p pop65p popurban medage
death
##   <chr>      <dbl+lb>  <dbl>  <dbl>   <dbl>  <dbl>  <dbl>    <dbl>  <dbl>
<dbl>
## 1 Alabama    3 [Sout… 3.89e6 2.96e5  865836 2.73e6 4.40e5  2337713   29.3
35305
## 2 Alaska     4 [West] 4.02e5 3.89e4   91796 2.71e5 1.15e4   258567   26.1
1604
## 3 Arizona    4 [West] 2.72e6 2.14e5  577604 1.93e6 3.07e5  2278728   29.2
21226
## 4 Arkansas   3 [Sout… 2.29e6 1.76e5  495782 1.62e6 3.12e5  1179556   30.6
22676
## 5 California 4 [West] 2.37e7 1.71e6 4680558 1.73e7 2.41e6 21607606   29.9
186428
## 6 Colorado   4 [West] 2.89e6 2.16e5  592318 2.08e6 2.47e5  2329869   28.6
18925
## # … with 2 more variables: marriage <dbl>, divorce <dbl>
```

¿Cuáles son las poblaciones totales de las regiones censales?:

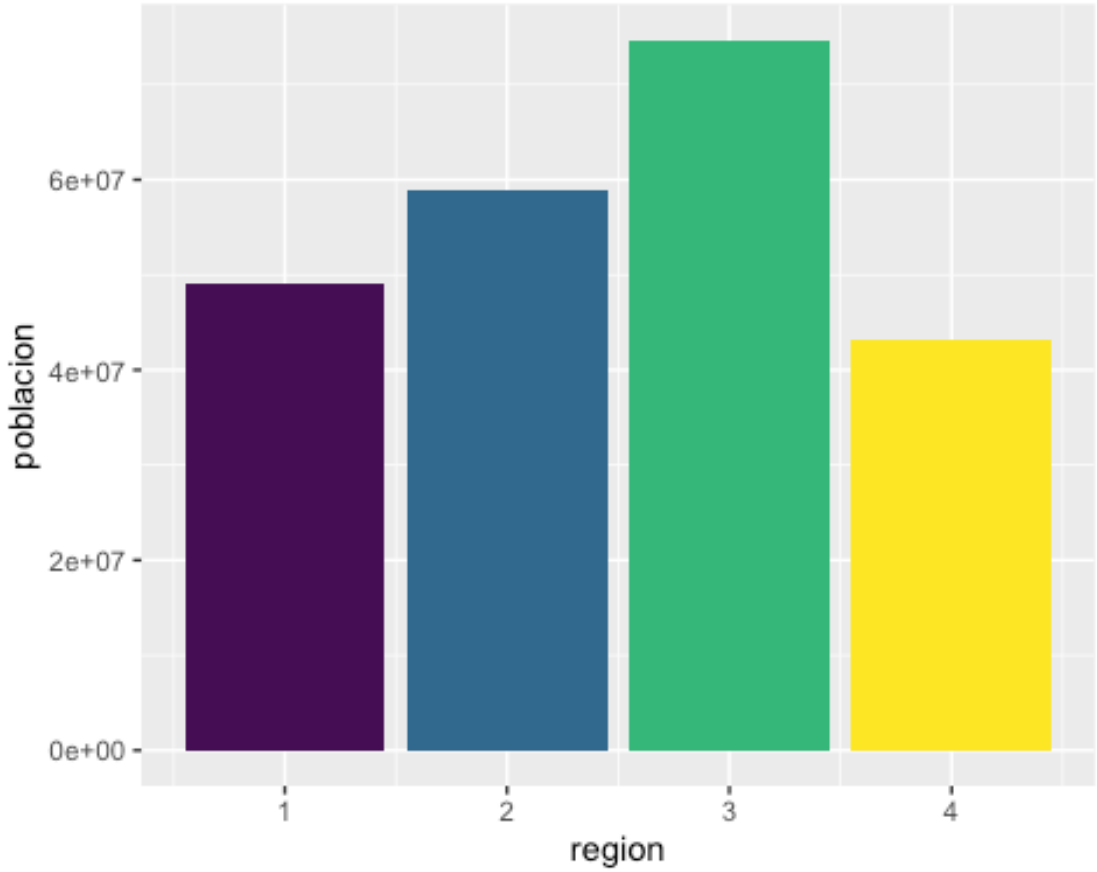
```
census %>% group_by(region) %>% summarise( poblacion=sum(pop) ) -> censoT
head(censoT)
```

```
## # A tibble: 4 × 2
##        region poblacion
##     <dbl+lbl>     <dbl>
## 1 1 [NE]       49135283
## 2 2 [N Cntrl]  58865670
## 3 3 [South]    74734029
## 4 4 [West]     43172490
```

Representa esas poblaciones totales en un diagrama de barras (una barra por región censal):

```
library(viridisLite)
ggplot(censoT, aes(region, poblacion)) +
  geom_col(fill=viridis(4))

## Don't know how to automatically pick scale for object of type
haven_labelled/vctrs_vctr/double. Defaulting to continuous.
```




Ordena los estados por población, de mayor a menor:

```
census %>% arrange(desc(pop))

## # A tibble: 50 × 12
##    state       region    pop poplt5 pop5_17 pop18p pop65p popurban medage
death
##    <chr>    <dbl+lbl>  <dbl>  <dbl>   <dbl>  <dbl>  <dbl>    <dbl>  <dbl>
<dbl>
##  1 Califor… 4 [West]  2.37e7 1.71e6 4680558 1.73e7 2.41e6 21607606   29.9
186428
##  2 New York 1 [NE]    1.76e7 1.14e6 3551938 1.29e7 2.16e6 14858068   31.9
171769
##  3 Texas    3 [South] 1.42e7 1.17e6 3137045 9.92e6 1.37e6 11333017   28.2
108019
##  4 Pennsyl… 1 [NE]    1.19e7 7.47e5 2375838 8.74e6 1.53e6  8220851   32.1
123261
```

```
##  5 Illinois 2 [N Cnt… 1.14e7 8.42e5 2400796 8.18e6 1.26e6  9518039   29.9
102230
##  6 Ohio     2 [N Cnt… 1.08e7 7.87e5 2307170 7.70e6 1.17e6  7918259   29.9
98268
##  7 Florida  3 [South] 9.75e6 5.70e5 1789412 7.39e6 1.69e6  8212385   34.7
104190
##  8 Michigan 2 [N Cnt… 9.26e6 6.85e5 2066873 6.51e6 9.12e5  6551551   28.8
75102
##  9 New Jer… 1 [NE]    7.36e6 4.63e5 1527572 5.37e6 8.60e5  6557377   32.2
68762
## 10 N. Caro… 3 [South] 5.88e6 4.04e5 1253659 4.22e6 6.03e5  2822852   29.6
48426
## # … with 40 more rows, and 2 more variables: marriage <dbl>, divorce <dbl>
```

Crea una nueva variable que contenga la tasa de divorcios/matrimonios para cada estado.

```
census %>% summarise( state, rateDivMa=divorce/marriage ) %>%
arrange(rateDivMa)
```

```
## # A tibble: 50 × 2
##    state         rateDivMa
##    <chr>             <dbl>
##  1 Nevada            0.121
##  2 S. Carolina       0.252
##  3 S. Dakota         0.319
##  4 N. Dakota         0.351
##  5 Pennsylvania      0.373
##  6 Hawaii            0.374
##  7 Maryland          0.378
##  8 Massachusetts     0.386
##  9 Virginia          0.392
## 10 Minnesota         0.408
## # … with 40 more rows
```

Crear la tabla con estado, edad mediana y proporción de adultos:

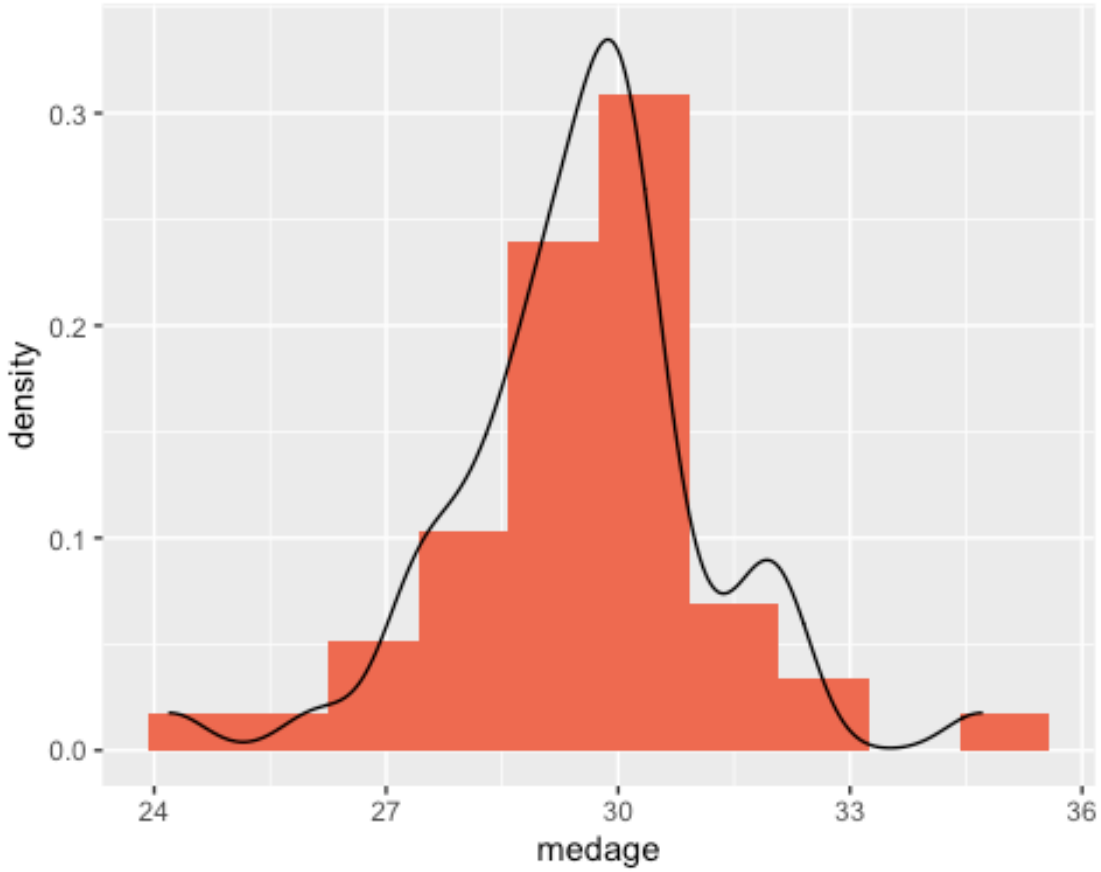
```
census %>%
  summarise(Estado=state ,Prop18=pop18p/pop, EdadMediana=medage) %>%
  arrange(EdadMediana) %>% head(10)
```

```
## # A tibble: 10 × 3
##    Estado      Prop18 EdadMediana
##    <chr>        <dbl>       <dbl>
##  1 Utah         0.630        24.2
##  2 Alaska       0.675        26.1
##  3 Wyoming      0.690        27.1
##  4 Louisiana    0.684        27.4
##  5 New Mexico   0.679        27.4
##  6 Idaho        0.675        27.6
##  7 Mississippi  0.677        27.7
##  8 S. Carolina  0.698        28.1
```

```
##  9 Texas        0.697        28.2
## 10 N. Dakota    0.707        28.3
```

Histograma y curva de densidad de la variable `medage`

```
ggplot(data=census)+geom_histogram(mapping =
aes(x=medage,y=stat(density)),bins=10,fill="coral2")+
geom_density(mapping = aes(medage))
```




## Ejercicio 1

### *Introducción*

Empezaremos cargando el fichero de datos *cholesterol.csv* y creamos el *data.frame* llamado `chlstrl`.

```
chlstrl <- read_csv("./data/cholesterol.csv")
```

Para obtener información básica sobre el conjunto de datos como cuantos registros tiene, el tipo de variables, el nombre de las columnas, el orden de magnitud de los registros podemos usar el comando `str()`.

```
str(chlstrl)
```

```
## spec_tbl_df [403 × 7] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ chol  : num [1:403] 203 165 228 78 249 248 195 227 177 263 ...
##  $ age   : num [1:403] 46 29 58 67 64 34 30 37 45 55 ...
##  $ gender: chr [1:403] "female" "female" "female" "male" ...
##  $ height: num [1:403] 62 64 61 67 68 71 69 59 69 63 ...
##  $ weight: num [1:403] 121 218 256 119 183 190 191 170 166 202 ...
##  $ waist : num [1:403] 29 46 49 33 44 36 46 34 34 45 ...
##  $ hip   : num [1:403] 38 48 57 38 41 42 49 39 40 50 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   chol = col_double(),
##   ..   age = col_double(),
##   ..   gender = col_character(),
##   ..   height = col_double(),
##   ..   weight = col_double(),
##   ..   waist = col_double(),
##   ..   hip = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

En cuanto a la comprobación de datos de la tabla, debemos asegurarnos que no tenemos nigún registro vacio. El comando `anyNA()` nos dice que la respuesta a la pregunta de si tenemos observaciones vacias es TRUE.

```
anyNA(chlstrl)
```

```
## [1] TRUE
```

De hecho si aplicamos la función `is.na()` que nos devuelve las posiciones de las observaciones vacias junto con la función `sum()` (en R TRUE equivale a un 1 y FALSE a un 0 cuando sumamos), obtenemos el número de registros vacios.

```
sum(is.na(chlstrl))
```

```
## [1] 11
```

Por tanto cuando estemos trabajando con estos datos debemos quitar estas observaciones vacias. Otra forma de trabajar es quitarlas directamente de la tabla con el comando `na.omit()` pero en este caso hemos preferido no usarlo ya que quita la fila entera donde se encuentra la observación vacia y no queremos perder tantos datos.

### *Análisis exploratorio*

A continuación procederemos a realizar un análisis exploratorio de los tipos de variables de la tabla, cuantitativas y categóricas. Un ejemplo de variable cuantitativa es la columna `chol` cuyo mínimo y máximo es 78 y 443 respectivamente. Presenta una media y mediana de 207.85 y 204 y una desviación estándar muestral de 44.446.

```
min(chlstrl$chol, na.rm = TRUE)
```

```
## [1] 78
```

```r
max(chlstrl$chol, na.rm = TRUE)
```

```
## [1] 443
```

```r
mean(chlstrl$chol, na.rm = TRUE)
```

```
## [1] 207.8458
```

```r
median(chlstrl$chol, na.rm = TRUE)
```

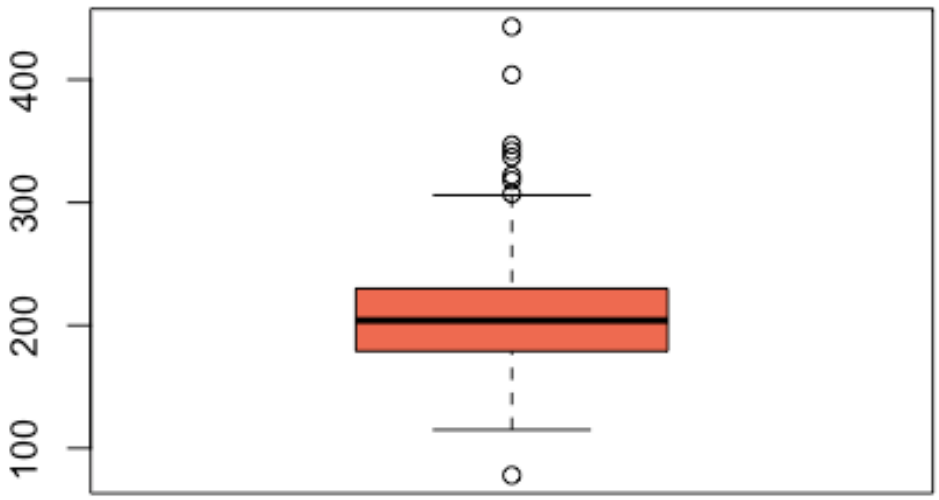
```
## [1] 204
```

```r
sd(chlstrl$chol, na.rm = TRUE)
```

```
## [1] 44.44556
```

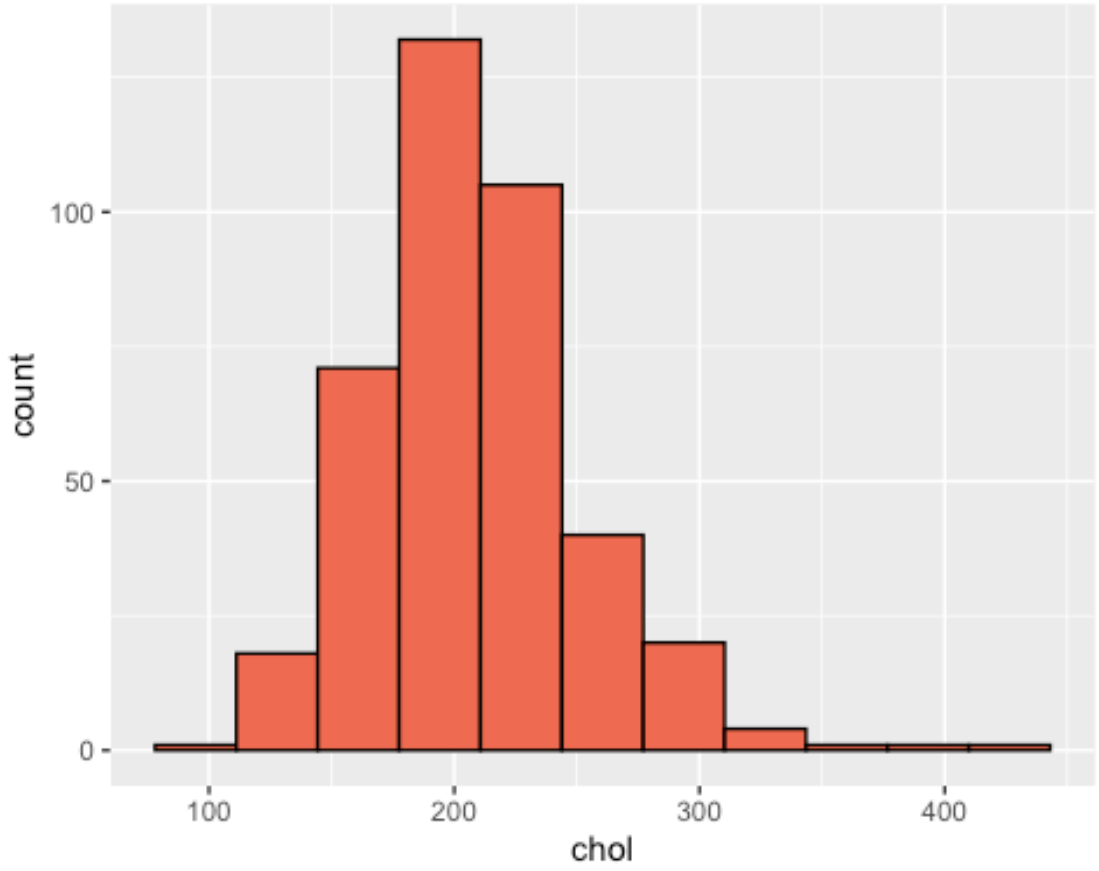
Se puede resumir gráficamente todas estas variables estadísticas en un diagrama de cajas, donde se aprecia la mediana, el primer cuartil, el tercer cuartil y los datos típicos y atípicos:

```r
bxp_cty = boxplot(na.omit(chlstrl$chol), col="coral2")
```



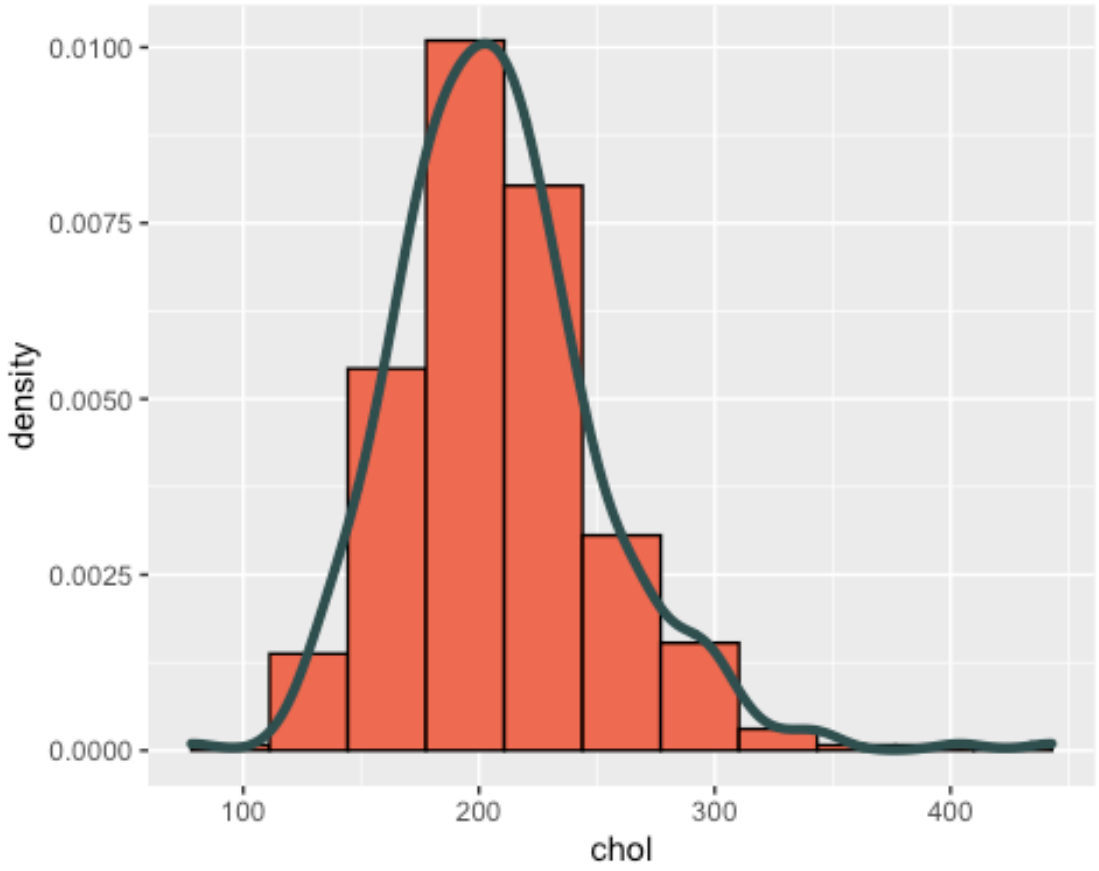

Si representamos en un histograma la tabla de frecuencias absolutas obtenida del colesterol de la muestra de pacientes obtenemos:

```
cortes = seq(min(chlstrl$chol,na.rm = TRUE), max(chlstrl$chol,na.rm = TRUE),
length.out = 12)
ggplot(data = na.omit(chlstrl)) +
  geom_histogram(mapping = aes(chol), breaks = cortes,
                 fill = "coral2", color="black")
```



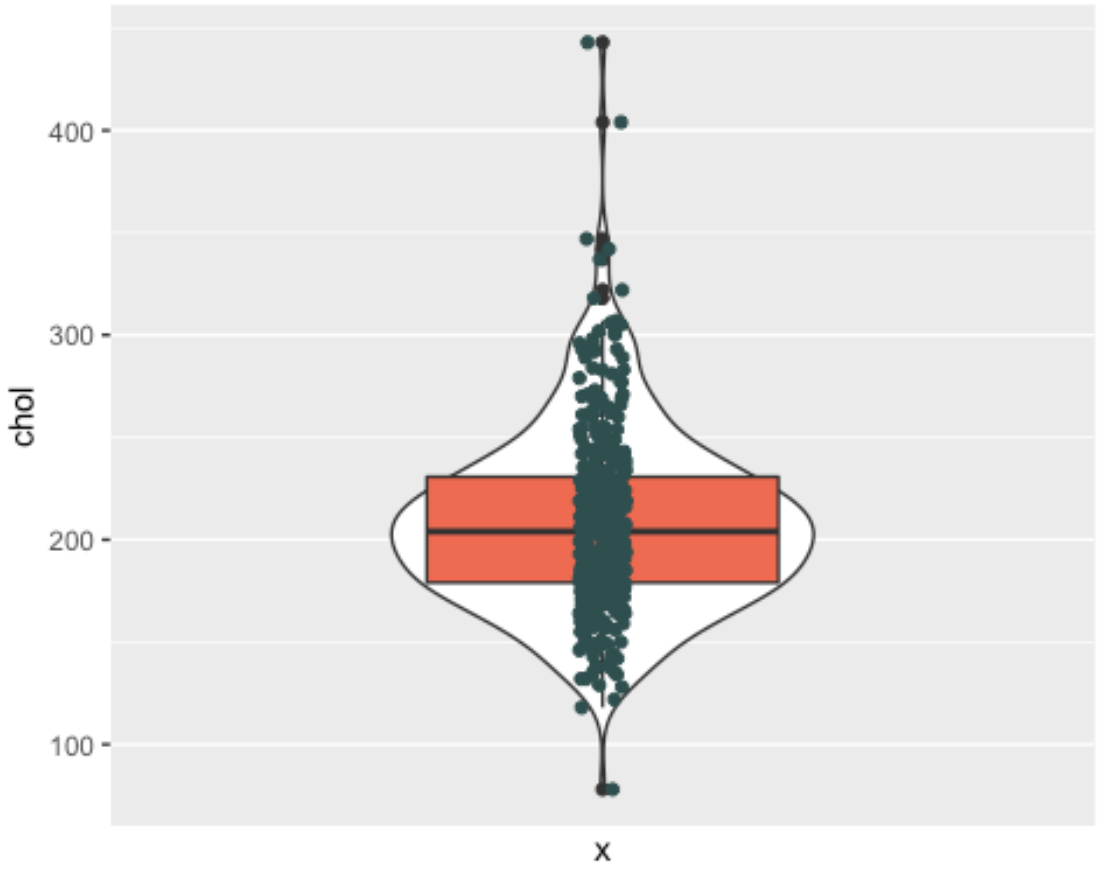

Mientras que si representamos de forma conjunta la curva de densidad junto con el histograma (pero representando las frecuencias relativas) tenemos:

```
ggplot(na.omit(chlstrl), aes(x = chol)) +
  geom_histogram(aes(x = chol,y=stat(density)),
                 breaks = cortes, fill = "coral2", color="black")  +
  geom_density(mapping = aes(chol),color="darkslategrey", size=1.5)
```




Por último, para terminar de realizar el análisis exploratorio, realizamos un *violinplot*, en el que se nos brinda de información del un diagrama de cajas además de disponer de curva de densidad y la diospersión de los puntos:

```
ggplot(na.omit(chlstrl)) +
  geom_violin(mapping = aes(x=0, y = chol)) + scale_x_discrete(breaks = c())
+
  geom_boxplot(mapping = aes(y = chol), fill="coral2") +
  geom_jitter(aes(x=0, y = chol),position = position_jitter(w=0.05, h= 0),
col="darkslategrey")
```




Por otro lado tenemos como ejemplo de una variable categórica o factor es la columna `gender`. Por defecto, cuando hemos importado la tabla, la columna `gender` se ha guardado como `string`. Por tanto primero debemos cambiar el tipo de la columna `gender` a factor.

```
chlstrl$gender=factor(chlstrl$gender)

class(chlstrl$gender)

## [1] "factor"
```

Para saber cuántos hombres y mujeres en la muestra usamos la tabla de frecuencias absolutas:

```
table(chlstrl$gender)

##
## female   male
##    234    169
```

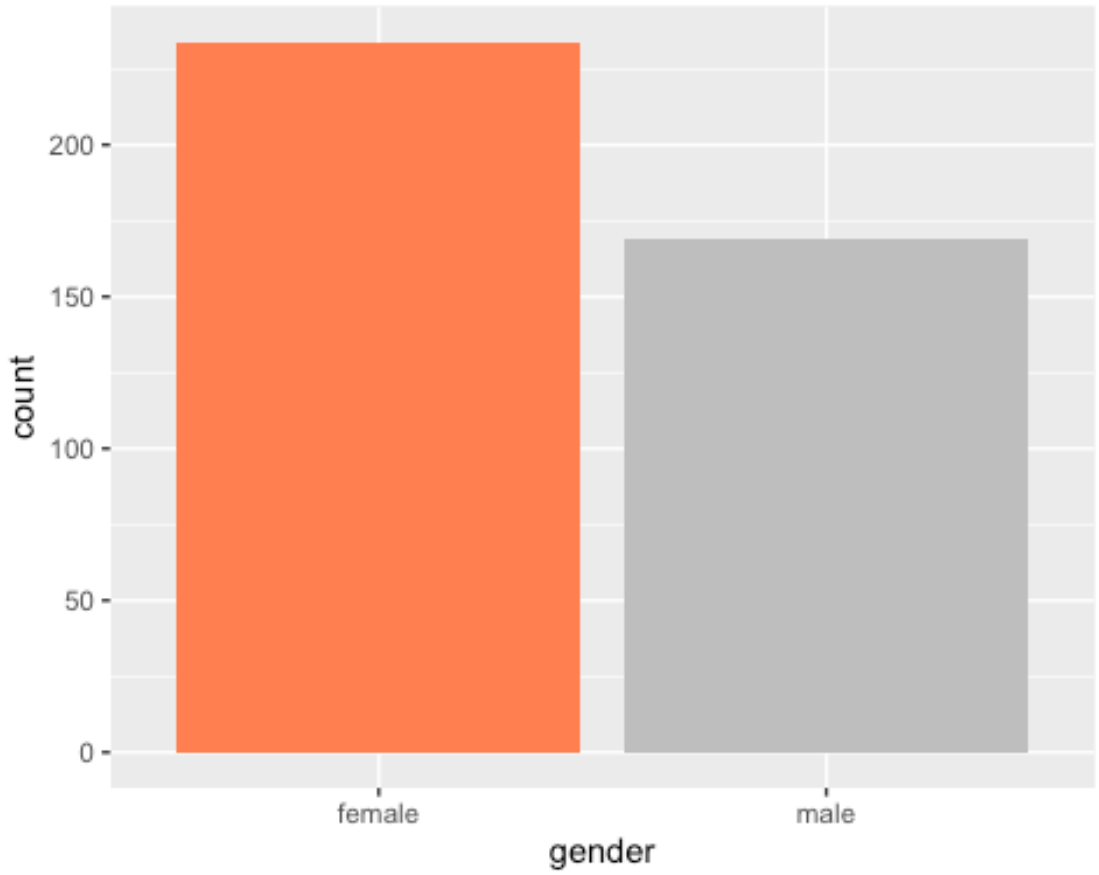
o una tabla de frecuencias relativas, que nos dice el porcentaje de hombres y mujeres en tanto por 1. Esto se debe a que trabajamos con factores dicotómicos.

```
prop.table(table(chlstrl$gender))
```

```
##
##    female      male
## 0.5806452 0.4193548
```

Podemos usar un diagrama de barras para representar una tabla de frecuencias absolutas:

```
ggplot(chlstrl) +
  geom_bar(mapping = aes(x = gender), fill= c("coral","grey"))
```




Dado que estamos interesados en trabajar en el sistema internacional, SI, realizamos el siguiente comando para cambiar las unidades de la altura, `height`, y del peso, `weight`. Con `mutate()` reemplazamos las columnas `height` y `weight` por las versiones en el Sistema Internacional.

```
chlstrl %>% mutate(height=height*0.0254,weight=weight*0.454) -> chlstrlSI

head (chlstrlSI)

## # A tibble: 6 × 7
##    chol   age gender height weight waist   hip
##   <dbl> <dbl> <fct>   <dbl>  <dbl> <dbl> <dbl>
## 1   203    46 female   1.57   54.9    29    38
## 2   165    29 female   1.63   99.0    46    48
## 3   228    58 female   1.55  116.     49    57
## 4    78    67 male     1.70   54.0    33    38
```

```
## 5   249    64 male     1.73   83.1    44    41
## 6   248    34 male     1.80   86.3    36    42
```

Usando el comando `mutate()` creamos la columna `BMI` (ya que no existe inicialmente la columna `BMI`, al usar `mutate()` se crea)

```
chlstrlSI %>%
  mutate("BMI" = weight/(height)^2) -> chlstrlSI
head (chlstrlSI)
```

```
## # A tibble: 6 × 8
##    chol   age gender height weight waist   hip   BMI
##   <dbl> <dbl> <fct>   <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1   203    46 female   1.57   54.9    29    38  22.2
## 2   165    29 female   1.63   99.0    46    48  37.5
## 3   228    58 female   1.55  116.     49    57  48.4
## 4    78    67 male     1.70   54.0    33    38  18.7
## 5   249    64 male     1.73   83.1    44    41  27.8
## 6   248    34 male     1.80   86.3    36    42  26.5
```

Usando el comando `cut()` creamos un vector de factores, `ageGroup`, dividiendo las edades en tres grupos.

```
ageGroup=cut(chlstrlSI$age, breaks = seq(10,100,30))
head(ageGroup)
```

```
## [1] (40,70] (10,40] (40,70] (40,70] (40,70] (10,40]
## Levels: (10,40] (40,70] (70,100]
```

Añadimos este vector a la tabla `chlstrlSI`:

```
chlstrlSI %>% mutate(ageGroup) -> chlstrlSI
head (chlstrlSI)
```

```
## # A tibble: 6 × 9
##    chol   age gender height weight waist   hip   BMI ageGroup
##   <dbl> <dbl> <fct>   <dbl>  <dbl> <dbl> <dbl> <dbl> <fct>
## 1   203    46 female   1.57   54.9    29    38  22.2 (40,70]
## 2   165    29 female   1.63   99.0    46    48  37.5 (10,40]
## 3   228    58 female   1.55  116.     49    57  48.4 (40,70]
## 4    78    67 male     1.70   54.0    33    38  18.7 (40,70]
## 5   249    64 male     1.73   83.1    44    41  27.8 (40,70]
## 6   248    34 male     1.80   86.3    36    42  26.5 (10,40]
```

Para saber la media del nivel de colesterol y de BMI de las mujeres en cada uno de los grupos de edad. Usamos `group_by()` para agrupar por grupos de edad, `group_by`, el comando `filter()` para decir que sean mujeres, y con el comando `summarise()` creamos un nuevo *data.frame* donde calculamos la media del colesterol y de la media.

```
chlstrlSI %>% group_by(ageGroup) %>% filter(gender=="female")  %>%
  summarise(MediaChol=mean(chol,na.rm = TRUE), MediaBMI=mean(BMI,na.rm =
TRUE))
```

```
## # A tibble: 3 × 3
##   ageGroup MediaChol MediaBMI
##   <fct>        <dbl>    <dbl>
## 1 (10,40]       189.     30.5
## 2 (40,70]       221.     30.3
## 3 (70,100]      230.     29.4
```

### Ejercicio 2

En primer lugar creamos el vector x de números enteros no nulos dado como ejemplo y un vector y que genera vectores aleatorios enteros no nulos.

```
x=c(-12,-19,9,-13,-14,-17,8,-19,-10)
y=sample(c(-20:-1,1:20),9,replace = TRUE)

## [1] -12 -19   9 -13 -14 -17

y=sample(c(-20:-1,1:20),9,replace = TRUE)

## [1] -13   7   6  19  14  -7
```

Ahora, creamos una función que calcula cuantos cambios de signo tiene el vector:

```
cambiosSigno=function(x){
i=0
  for(k in seq(length(x)-1)){
    if( x[k]*x[k+1]<0 ){
      i=i+1
    }
  }
return(i)
}
```

Y la funciónb que calcula en que posiciones se han producido los cambios de signo y devuelve un mensaje cuando no se ha producido ningún cambio:

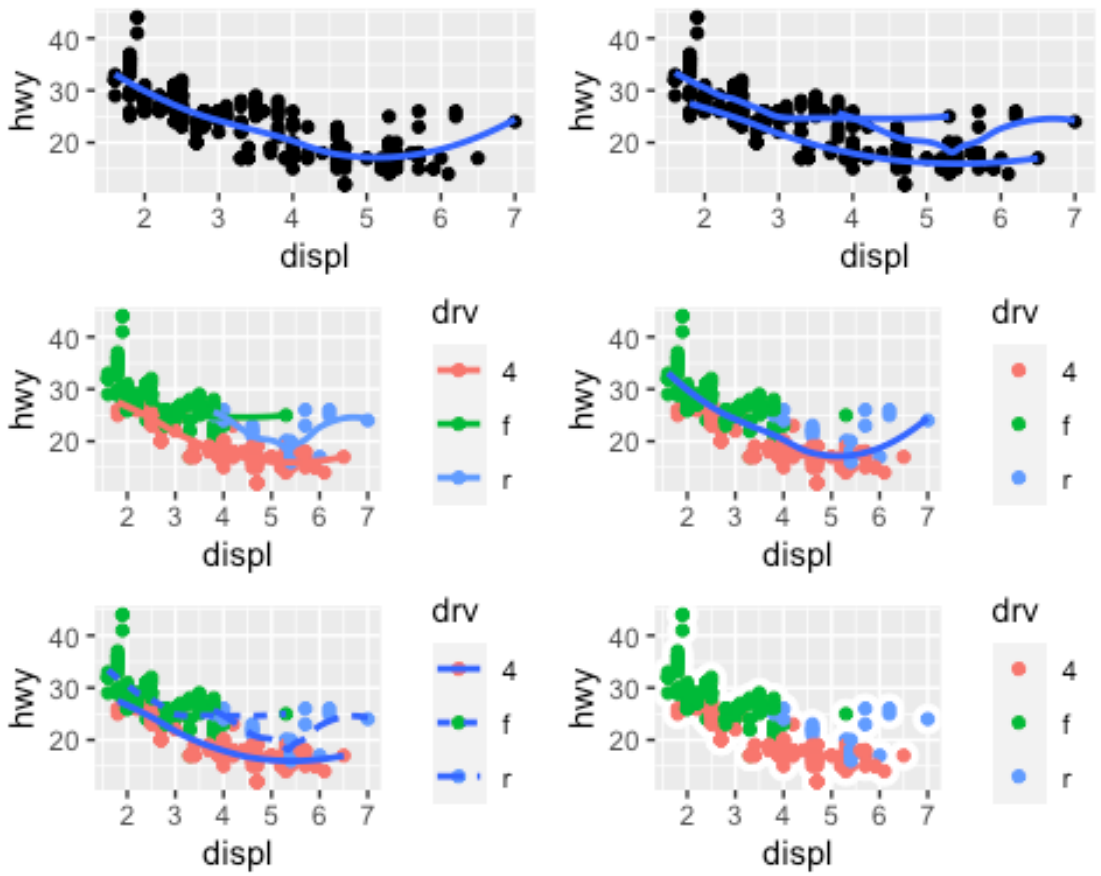
```
cambiosSignoPos=function(x){
pos=c()
  for(k in seq(length(x)-1)){
    if( x[k]*x[k+1]<0 ){
      pos=append(pos,k+1)
    }
  }
  if( is.null(pos) == TRUE){
    print("No hay ningún cambio de signo")
    }else{
      return(pos)
    }
}
```

### Ejercicio 3

Queremos replicar las gráficas que aparecen en la sección 3 y 5 del libro R for Data Science. Guardamos cada gráfica en una variable y luego usamos el comando `grid.arrange`.

```
library(gridExtra)

ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point() +
  geom_smooth(se = FALSE,method = 'loess',formula =y ~ x) -> g1
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, group = drv)) +
  geom_point() +
  geom_smooth(se = FALSE,method = 'loess',formula =y ~ x) -> g2
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, colour = drv)) +
  geom_point() +
  geom_smooth(se = FALSE,method = 'loess',formula =y ~ x) -> g3
ggplot() +
  geom_point(data = mpg, mapping = aes(x = displ, y = hwy, colour = drv)) +
  geom_smooth(data = mpg, mapping = aes(x = displ, y = hwy),se = FALSE,method
= 'loess',formula =y ~ x) -> g4
ggplot() +
  geom_point(data = mpg, mapping = aes(x = displ, y = hwy, colour = drv)) +
  geom_smooth(data = mpg, mapping = aes(x = displ, y = hwy, linetype =
drv),se = FALSE,method = 'loess',formula =y ~ x) -> g5
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_point(size = 4, color = "white") +
  geom_point(aes(colour = drv)) -> g6
grid.arrange(g1, g2, g3, g4,g5,g6, nrow = 3)
```




Ahora realizaremos una serie de consultas que realizaremos con el comando `filter()`. Primero cargamos la libreria necesaria:

```
library(nycflights13)
```

Vuelos que tengan retraso en la hora de llegada de dos horas o más:

```
filter(flights, arr_delay>=120 )
```

```
## # A tibble: 10,200 × 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
<int>
##  1  2013     1     1      811            630       101     1047
830
##  2  2013     1     1      848           1835       853     1001
1950
##  3  2013     1     1      957            733       144     1056
853
##  4  2013     1     1     1114            900       134     1447
1222
##  5  2013     1     1     1505           1310       115     1638
1431
```

```
##  6  2013     1     1     1525           1340       105     1831
1626
##  7  2013     1     1     1549           1445        64     1912
1656
##  8  2013     1     1     1558           1359       119     1718
1515
##  9  2013     1     1     1732           1630        62     2028
1825
## 10  2013     1     1     1803           1620       103     2008
1750
## # … with 10,190 more rows, and 11 more variables: arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour
<dttm>
```

Vuelos que volaron a Houston:

```
filter(flights, dest == "IAH" | dest == "HOU")
```

```
## # A tibble: 9,313 × 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
<int>
##  1  2013     1     1      517            515         2      830
819
##  2  2013     1     1      533            529         4      850
830
##  3  2013     1     1      623            627        -4      933
932
##  4  2013     1     1      728            732        -4     1041
1038
##  5  2013     1     1      739            739         0     1104
1038
##  6  2013     1     1      908            908         0     1228
1219
##  7  2013     1     1     1028           1026         2     1350
1339
##  8  2013     1     1     1044           1045        -1     1352
1351
##  9  2013     1     1     1114            900       134     1447
1222
## 10  2013     1     1     1205           1200         5     1503
1505
## # … with 9,303 more rows, and 11 more variables: arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour
<dttm>
```

Vuelos cuya operadora fue United, American o Delta:

```
filter(flights, carrier == "UA" | carrier == "AA" | carrier == "DL")
```

```
## # A tibble: 139,504 × 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
<int>
##  1  2013     1     1      517            515         2      830
819
##  2  2013     1     1      533            529         4      850
830
##  3  2013     1     1      542            540         2      923
850
##  4  2013     1     1      554            600        -6      812
837
##  5  2013     1     1      554            558        -4      740
728
##  6  2013     1     1      558            600        -2      753
745
##  7  2013     1     1      558            600        -2      924
917
##  8  2013     1     1      558            600        -2      923
937
##  9  2013     1     1      559            600        -1      941
910
## 10  2013     1     1      559            600        -1      854
902
## # … with 139,494 more rows, and 11 more variables: arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour
<dttm>
```

Vuelos que se realizaron en verano:

```
filter(flights, month == 6 | month == 7 | month == 8)
```

```
## # A tibble: 86,995 × 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
<int>
##  1  2013     6     1        2           2359         3      341
350
##  2  2013     6     1      451            500        -9      624
640
##  3  2013     6     1      506            515        -9      715
800
##  4  2013     6     1      534            545       -11      800
829
##  5  2013     6     1      538            545        -7      925
922
```

```
##  6  2013     6     1      539            540        -1      832
840
##  7  2013     6     1      546            600       -14      850
910
##  8  2013     6     1      551            600        -9      828
850
##  9  2013     6     1      552            600        -8      647
655
## 10  2013     6     1      553            600        -7      700
711
## # … with 86,985 more rows, and 11 more variables: arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour
<dttm>
```

No salieron tarde pero llegaron más de dos horas tarde:

```
filter(flights, dep_delay<=0 , arr_delay>=120)
```

```
## # A tibble: 29 × 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
<int>
##  1  2013     1    27     1419           1420        -1     1754
1550
##  2  2013    10     7     1350           1350         0     1736
1526
##  3  2013    10     7     1357           1359        -2     1858
1654
##  4  2013    10    16      657            700        -3     1258
1056
##  5  2013    11     1      658            700        -2     1329
1015
##  6  2013     3    18     1844           1847        -3       39
2219
##  7  2013     4    17     1635           1640        -5     2049
1845
##  8  2013     4    18      558            600        -2     1149
850
##  9  2013     4    18      655            700        -5     1213
950
## 10  2013     5    22     1827           1830        -3     2217
2010
## # … with 19 more rows, and 11 more variables: arr_delay <dbl>, carrier
<chr>,
## #   flight <int>, tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>,
## #   distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

Vuelos que se retrasaron por lo menos una hora en la salida, pero recuperaron media hora:

```
filter(flights, dep_delay-arr_delay<=30, dep_delay>=60)
```

```
## # A tibble: 24,958 × 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
<int>
##  1  2013     1     1      811            630       101     1047
830
##  2  2013     1     1      826            715        71     1136
1045
##  3  2013     1     1      848           1835       853     1001
1950
##  4  2013     1     1      957            733       144     1056
853
##  5  2013     1     1     1114            900       134     1447
1222
##  6  2013     1     1     1120            944        96     1331
1213
##  7  2013     1     1     1301           1150        71     1518
1345
##  8  2013     1     1     1337           1220        77     1649
1531
##  9  2013     1     1     1400           1250        70     1645
1502
## 10  2013     1     1     1505           1310       115     1638
1431
## # … with 24,948 more rows, and 11 more variables: arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour
<dttm>
```

Salieron entre medianoche y las 6 AM:

```
filter(flights, dep_time>=0 & dep_time<=600)
```

```
## # A tibble: 9,344 × 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
<int>
##  1  2013     1     1      517            515         2      830
819
##  2  2013     1     1      533            529         4      850
830
##  3  2013     1     1      542            540         2      923
850
##  4  2013     1     1      544            545        -1     1004
1022
##  5  2013     1     1      554            600        -6      812
837
```

```
##  6  2013     1     1      554            558        -4      740
728
##  7  2013     1     1      555            600        -5      913
854
##  8  2013     1     1      557            600        -3      709
723
##  9  2013     1     1      557            600        -3      838
846
## 10  2013     1     1      558            600        -2      753
745
## # … with 9,334 more rows, and 11 more variables: arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour
<dttm>
```