# Tarea 1

Álvaro Francisco Ruiz Cornejo

15/9/2021

## Ejercicio 1. Análisis exploratorio de un conjunto de datos y operaciones con dplyr.

En primer lugar, comprobamos que los datos han sido cargados correctamente en un dataframe de R, mostrando únicamente las primeras filas.

```
head(chlstrl)
```

```
##   chol age gender height weight waist hip
## 1  203  46 female     62    121    29  38
## 2  165  29 female     64    218    46  48
## 3  228  58 female     61    256    49  57
## 4   78  67   male     67    119    33  38
## 5  249  64   male     68    183    44  41
## 6  248  34   male     71    190    36  42
```

- **¿Cuántas observaciones tiene?**

```
dim(chlstrl) # Número de filas y columnas, respectivamente
```

```
## [1] 403   7
```

```
nrow(chlstrl) # Número de filas
```

```
## [1] 403
```

```
ncol(chlstrl) # Número de columnas
```

```
## [1] 7
```

- **¿Cúales son las variables y de qué tipo?**

```
summary.default(chlstrl)
```

```
##        Length Class  Mode
## chol   403    -none- numeric
## age    403    -none- numeric
## gender 403    -none- character
## height 403    -none- numeric
## weight 403    -none- numeric
## waist  403    -none- numeric
## hip    403    -none- numeric
```

- **¿Hay datos ausentes?¿Cuántos?¿En qué variables y posición están?**

```
any(is.na(chlstrl)) # ¿Hay algún dato ausente?
```

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

```
sum(is.na(chlstrl)) # ¿Cuántos datos ausentes hay?
```

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

```
apply(is.na(chlstrl),2,which) # ¿Dónde están los datos ausentes?
```

```
## $chol
## [1] 28
##
## $age
## integer(0)
##
## $gender
## integer(0)
##
## $height
## [1]  64  87 196 232 318
##
## $weight
## [1] 162
##
## $waist
## [1] 337 394
##
## $hip
## [1] 337 394
```

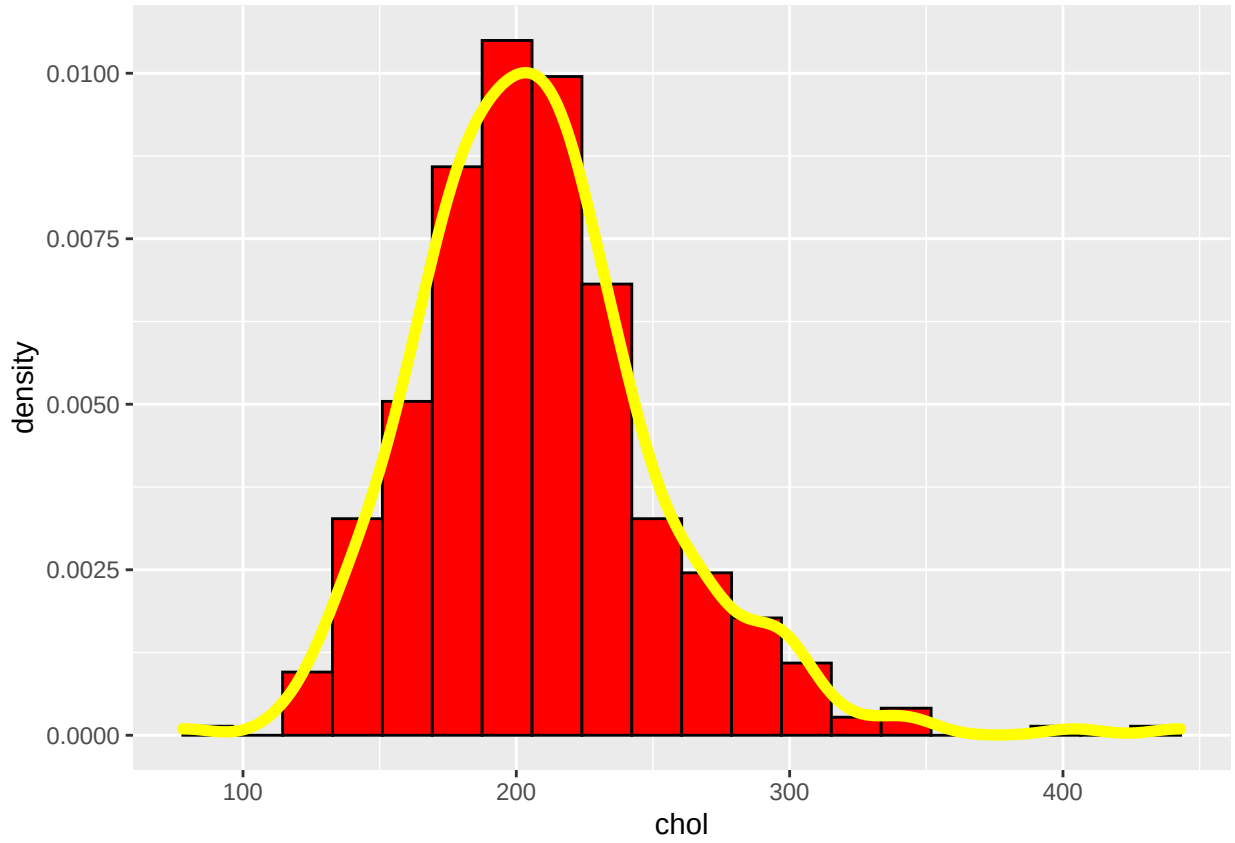
- **El análisis exploratorio (numérico y gráfico) debe cubrir todos los tipos de variable de la tabla. Es decir, que al menos debes estudiar una variable por cada tipo de variable presente en la tabla.**

A continuación se va a realizar el estudio de una variable de tipo numérico, por ejemplo, el colesterol. Es una variable continua sobre la que vamos a realizar un estudio numérico básico y la presentación de una serie de gráficas acorde a la información contenida en la variable.

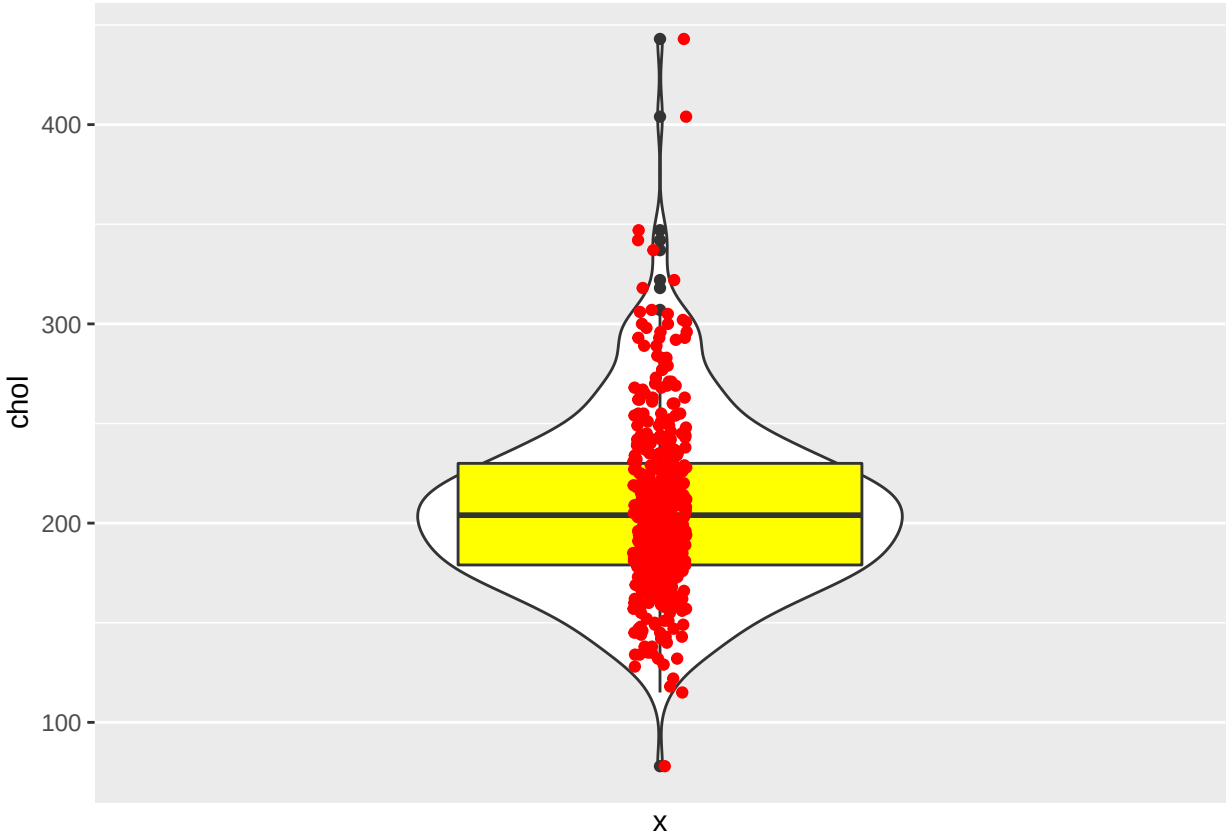
```
summary(chlstrl$chol) # Información numérica básica de la variable
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    78.0   179.0   204.0   207.8   230.0   443.0       1
```

```
cortes = seq(min(chlstrl$chol, na.rm = TRUE), max(chlstrl$chol, na.rm = TRUE),
             length.out = 21)
ggplot(data = chlstrl, mapping = aes(x = chol)) +
  geom_histogram(aes(y = stat(density)), breaks = cortes, color = "black", fill = "red") +
  geom_density(color = "yellow", size = 2)
```




```
ggplot(data = chlstrl) +
  geom_violin(mapping = aes(x = 0, y = chol)) +
  scale_x_discrete(breaks = c()) +
  geom_boxplot(mapping = aes(y = chol), fill = "yellow") +
  geom_jitter(aes(x = 0, y = chol), position = position_jitter(w = 0.05, h = 0),
              col = "red")
```




Por último, vamos a estudiar la variable *gender* como variable de tipo factor, que puede tomar los valores male/female únicamente. Para ello, la convertiremos en una variable de este tipo y calcularemos tanto su tabla de frecuencias (relativas y absolutas) como un diagrama de barras que reflejen los resultados.

```
chlstrl$gender = factor(chlstrl$gender) # Conversión de la variable a tipo factor
```

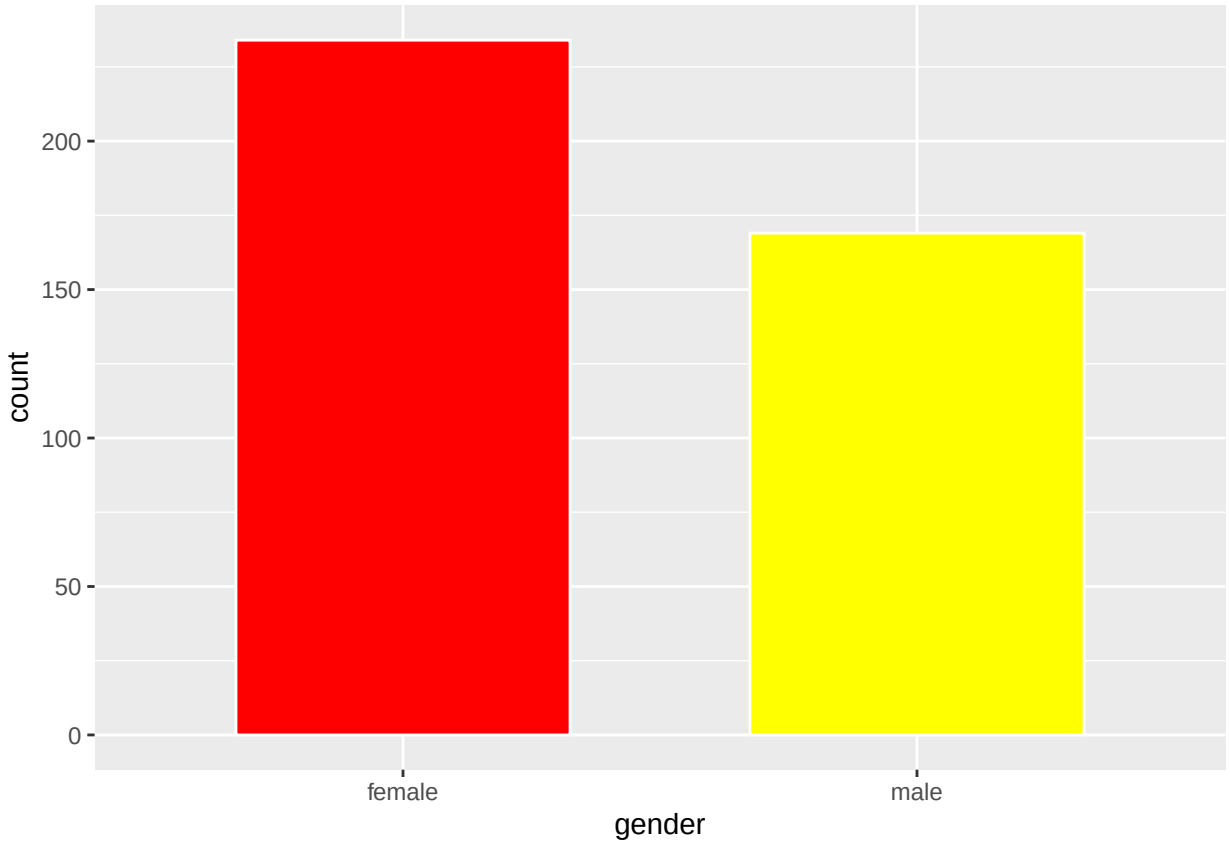
```
chlstrl %>%
  count(gender) # Tabla de frecuencias absolutas
```

```
##   gender   n
## 1 female 234
## 2   male 169
```

```
chlstrl %>%
  count(gender) %>%
  mutate(relFreq = prop.table(n), n = NULL) # Tabla de frecuencias relativas
```

```
##   gender   relFreq
## 1 female 0.5806452
## 2   male 0.4193548
```

```
ggplot(data = chlstrl) +
  geom_bar(mapping = aes(x = gender), colour = "white", fill = c("red", "yellow"),
           width = 0.65)
```




- **Los valores de height y weight están en pulgadas (inches) y libras (pounds) respectivamente. Una ibra son 0.454kg y una pulgada son 0.0254m. Usa dplyr para convertir esas columnas a metros y kilogramos respectivamente. Las nuevas columnas deben llamarse igual que las originales.**

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

```
chlstrl %>%
  head(10) # Mostramos únicamente las 10 primeras filas
```

```
##    chol age gender height  weight waist hip
## 1   203  46 female 1.5748  54.934    29  38
## 2   165  29 female 1.6256  98.972    46  48
## 3   228  58 female 1.5494 116.224    49  57
## 4    78  67   male 1.7018  54.026    33  38
## 5   249  64   male 1.7272  83.082    44  41
## 6   248  34   male 1.8034  86.260    36  42
```

```
## 7   195  30   male 1.7526  86.714    46  49
## 8   227  37   male 1.4986  77.180    34  39
## 9   177  45   male 1.7526  75.364    34  40
## 10  263  55 female 1.6002  91.708    45  50
```

- **Ahora usa esos valores de height y weight para añadir una nueva columna llamada BMI, que sea el resultado de dividir el peso entre la altura al cuadrado**

```
chlstrl <- chlstrl %>%
            mutate(BMI = weight/((height)^2))
```

```
chlstrl %>%
  head(10) # Mostramos únicamente las 10 primeras filas
```

```
##    chol age gender height  weight waist hip      BMI
## 1   203  46 female 1.5748  54.934    29  38 22.15085
## 2   165  29 female 1.6256  98.972    46  48 37.45286
## 3   228  58 female 1.5494 116.224    49  57 48.41375
## 4    78  67   male 1.7018  54.026    33  38 18.65459
## 5   249  64   male 1.7272  83.082    44  41 27.84977
## 6   248  34   male 1.8034  86.260    36  42 26.52316
## 7   195  30   male 1.7526  86.714    46  49 28.23083
## 8   227  37   male 1.4986  77.180    34  39 34.36634
## 9   177  45   male 1.7526  75.364    34  40 24.53569
## 10  263  55 female 1.6002  91.708    45  50 35.81448
```

- **Crea una nueva columna llamada ageGroup dividiendo la edad en los siguientes tres niveles: (10,40], (40,70], (70,100]**

```
chlstrl <- chlstrl %>%
              mutate(ageGroup = cut(age, breaks = c(10,40,70,100),
              labels = c("(10,40]", "(40,70]", "(70,100]"),
              include.lowest = FALSE, right = TRUE))
```

```
chlstrl %>%
  head(10) # Mostramos únicamente las 10 primeras filas
```

```
##    chol age gender height  weight waist hip      BMI ageGroup
## 1   203  46 female 1.5748  54.934    29  38 22.15085  (40,70]
## 2   165  29 female 1.6256  98.972    46  48 37.45286  (10,40]
## 3   228  58 female 1.5494 116.224    49  57 48.41375  (40,70]
## 4    78  67   male 1.7018  54.026    33  38 18.65459  (40,70]
## 5   249  64   male 1.7272  83.082    44  41 27.84977  (40,70]
## 6   248  34   male 1.8034  86.260    36  42 26.52316  (10,40]
## 7   195  30   male 1.7526  86.714    46  49 28.23083  (10,40]
## 8   227  37   male 1.4986  77.180    34  39 34.36634  (10,40]
## 9   177  45   male 1.7526  75.364    34  40 24.53569  (40,70]
## 10  263  55 female 1.6002  91.708    45  50 35.81448  (40,70]
```

- **Usando dplyr calcula cuántas observaciones hay en cada nivel de ageGroup (indicación: usa group_by).**

```
chlstrl %>%
  group_by(ageGroup) %>%
  count()
```

```
## # A tibble: 3 x 2
## # Groups:   ageGroup [3]
##   ageGroup     n
##   <fct>    <int>
## 1 (10,40]    160
## 2 (40,70]    207
## 3 (70,100]    36
```

- **Ahora, usando aquellas observaciones que corresponden a mujeres, ¿cuál es la media del nivel de colesterol y de BMI en cada uno de esos grupos de edad?**

```
(chlstrl %>%
  group_by(ageGroup) %>%
  filter(gender == "female") %>%
  summarise("media_chol" = mean(chol, na.rm = TRUE),
            "media_BMI" = mean(BMI, na.rm = TRUE)))
```

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

## Ejercicio 2. Funciones de R.

- **Crea una función de R llamada cambiosSigno que dado un vector x de números enteros no nulos, como -12, -19, 9, -13, -14, -17, 8, -19, -14, calcule cuántos cambios de signo ha habido. Es decir, cuántas veces el signo de un elemento es distinto del signo del elemento previo. Por ejemplo, en el vector anterior hay 4 cambios de signo (en las posiciones 3, 4, 7 y 8).**

La implementación en R de la función es la siguiente:

```
cambiosSigno = function(vector){
  numCambios = 0
  for(k in 2:length(vector)){
    if(vector[k]*vector[k-1] < 0){
      numCambios = numCambios + 1
```

```
    }
  }
  return(numCambios)
}
```

A partir del vector dado como ejemplo y haciendo una llamada a la función creada, obtenemos el número total de cambios de signo en dicho vector:

```
vector = c(-12,-19,9,-13,-14,-17,8,-19,-14)
cambiosSigno(vector)
```

```
## [1] 4
```

- **Modifica la función para que devuelva como resultado las posiciones donde hay cambios de signo. Llama cambiosSignoPos(x) a esa otra función. Por ejemplo, para el vector anterior el resultado de esta función sería [1] 3 4 7 8.**

De nuevo, la implentación en R sería la siguiente:

```
cambiosSignoPos = function(vector){
  listaCambios = c()
  for(k in 2:length(vector)){
    if(vector[k]*vector[k-1] < 0){
      listaCambios = append(listaCambios, k)
    }
  }
  return(listaCambios)
}
```

Y la lista con las posiciones en las que ha cambiado el signo:

```
vector = c(-12,-19,9,-13,-14,-17,8,-19,-14)
cambiosSignoPos(vector)
```

```
## [1] 3 4 7 8
```

- **También se valorará que incluyas en el código como usar sample para generar vectores aleatorios de 20 enteros no nulos (el vector debe poder tomar valores positivos y negativos).**

Se comprueba el funcionamiento de ambas funciones a partir de un vector generado aleatoriamente:

```
valores = c(-10:-1,1:10)
(rVector = sample(valores, size = 20, replace = TRUE)) # Vector generado aleatoriamente
```

```
##  [1] -3 -4  8 -5 -8  7 -6  7 10 10  3 -4 -9  9 -8  8 -1 -2 -4 -5
```

```r
cambiosSigno(rVector) # Número total de cambios de signo
```

```
## [1] 10
```

```r
cambiosSignoPos(rVector) # Lista de posiciones donde ha cambiado el signo
```

```
##  [1]  3  4  6  7  8 12 14 15 16 17
```

# Ejercicio 3. R4DS.

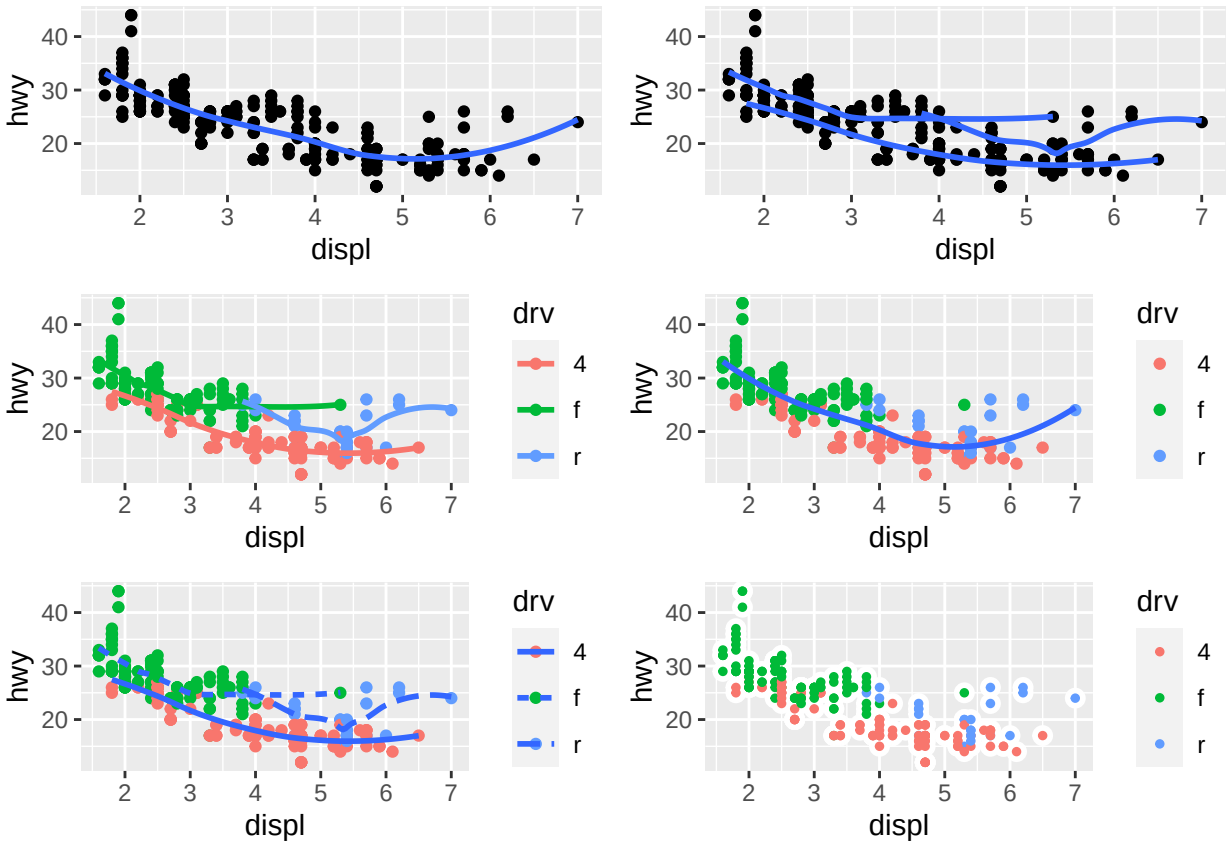
- **Haz el ejercicio 6 de la Sección 3.6.1 de R4DS.**

La forma de implementar las gráficas en código R es la siguiente:

```r
g1 <- ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point() +
  geom_smooth(se = FALSE)

g2 <- ggplot(data = mpg, mapping = aes(x = displ, y = hwy, group = drv)) +
  geom_point() +
  geom_smooth(se = FALSE)

g3 <- ggplot(data = mpg, mapping = aes(x = displ, y = hwy, colour = drv)) +
  geom_point() +
  geom_smooth(se = FALSE)

g4 <- 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)

g5 <- 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)

g6 <- ggplot(data = mpg, aes(x = displ, y = hwy)) +
  geom_point(size = 3, colour = "white") +
  geom_point(aes(colour = drv), size = 1)
```

El resultado de las seis gráficas, agrupadas en 3 filas se muestra a continuación:

```r
grid.arrange(g1,g2,g3,g4,g5,g6, nrow = 3)
```




- **Haz el ejercicio 1 de la Sección 5.2.4 de R4DS.**

Utilizamos la tabla flights de la librería nycflights13 para responder las siguientes cuestiones:

- Todos los vuelos que hayan tenido un retraso de dos o más horas:

```
names(flights)
```

```
##  [1] "year"           "month"          "day"            "dep_time"
##  [5] "sched_dep_time" "dep_delay"      "arr_time"       "sched_arr_time"
##  [9] "arr_delay"      "carrier"        "flight"         "tailnum"
## [13] "origin"         "dest"           "air_time"       "distance"
## [17] "hour"           "minute"         "time_hour"
```

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

```
## # A tibble: 10,200 x 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>
```

- Todos los vuelos que volaron a Houston (IAH o HOU):

```
flights %>%
  filter(dest %in% c('HOU','IAH'))
```

```
## # A tibble: 9,313 x 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>
```

- Todos los vuelos que fueron operados por “United”, “América” o “Delta”:

```
flights %>%
  filter(carrier %in% c('UA','AA','DL'))
```

```
## # A tibble: 139,504 x 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>
```

- Todos los vuelos que partieron en verano (julio, agosto o septiembre):

```
flights %>%
  filter(month %in% c(7:9))
```

```
## # A tibble: 86,326 x 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     7     1        1           2029       212      236           2359
##  2  2013     7     1        2           2359         3      344            344
##  3  2013     7     1       29           2245       104      151              1
##  4  2013     7     1       43           2130       193      322             14
##  5  2013     7     1       44           2150       174      300            100
##  6  2013     7     1       46           2051       235      304           2358
##  7  2013     7     1       48           2001       287      308           2305
##  8  2013     7     1       58           2155       183      335             43
##  9  2013     7     1      100           2146       194      327             30
## 10  2013     7     1      100           2245       135      337            135
## # ... with 86,316 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>
```

- Todos los vuelos que llegaron más de dos horas tarde pero que salieron a su hora:

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

```
## # A tibble: 29 x 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>
```

- Todos los vuelos que se retrasaron una hora pero que recuperaron 30 minutos durante el vuelo:

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

```
## # A tibble: 1,844 x 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     2205           1720       285       46           2040
##  2  2013     1     1     2326           2130       116      131             18
##  3  2013     1     3     1503           1221       162     1803           1555
##  4  2013     1     3     1839           1700        99     2056           1950
##  5  2013     1     3     1850           1745        65     2148           2120
##  6  2013     1     3     1941           1759       102     2246           2139
##  7  2013     1     3     1950           1845        65     2228           2227
##  8  2013     1     3     2015           1915        60     2135           2111
##  9  2013     1     3     2257           2000       177       45           2224
## 10  2013     1     4     1917           1700       137     2135           1950
## # ... with 1,834 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>
```

- Todos los vuelos que salieron entre medianoche y las seis de la mañana:

```
flights %>%
  filter(dep_time <= 600 | dep_time == 2400)
```

```
## # A tibble: 9,373 x 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 9,363 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>
```