

Tarea 1. Fundamentos Matemáticos.

Tarea 1

Mónica de Santos

Curso 2021-22. Última actualización: 2021-09-17

Instrucciones preliminares

- Empieza abriendo el proyecto de RStudio correspondiente a tu repositorio personal de la asignatura.
- En todas las tareas tendrás que repetir un proceso como el descrito en la sección *Repite los pasos Creando un fichero Rmarkdown para esta práctica* de la *Práctica00*. Puedes releer la sección *Practicando la entrega de las Tareas* de esa misma práctica para recordar el procedimiento de entrega.

Ejercicio 0

- Si no has hecho los *Ejercicios* de la *Práctica00* (págs. 12 y 13) hazlos ahora y añádelos a esta tarea. Si ya los has hecho y entregado a través de GitHub no hace falta que hagas nada.

```
library(tidyverse)
```

```
## -- Attaching packages ----- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.3      v purrr   0.3.4
## v tibble  3.1.2      v dplyr  1.0.6
## v tidyr   1.1.3      v stringr 1.4.0
## v readr   1.4.0      v forcats 0.5.1
```

```
## -- Conflicts ----- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()     masks stats::lag()
```

```
library(dplyr)
```

PRÁCTICA 0

Ejercicio 1:

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

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

#Tabla de frecuencias absolutas:

```
table(dado_honesto)
```

```
## dado_honesto
## 1 2 3 4 5 6
## 11 14 14 21 18 22
```

#Tabla de frecuencias relativas

```
prop.table(table(dado_honesto))
```

```
## dado_honesto
## 1 2 3 4 5 6
## 0.11 0.14 0.14 0.21 0.18 0.22
```

Ejercicio 2:

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

#Tabla de frecuencias absolutas:

```
table(dado_cargado)
```

```
## dado_cargado
## 1 2 3 4 5 6
## 22 9 19 16 10 24
```

#Tabla de frecuencias relativas

```
prop.table(table(dado_cargado))
```

```
## dado_cargado
## 1 2 3 4 5 6
## 0.22 0.09 0.19 0.16 0.10 0.24
```

Ejercicio 3:

#vectores con rep y seq

```
v1 = rep(c(4,3,2,1),c(4,4,4,4))
v2 = rep(seq(1:5), (c(1,2,3,4,5)))
v3 = rep(1:4,c(4))
```

Ejercicio 4:

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

```
## # A tibble: 33 x 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
## 7     8     9 pickup
## 8     8    11 pickup
## 9     8    11 pickup
## 10    8    12 pickup
## # ... with 23 more rows
```

Ejercicio 5:

```
library(haven)
census <- read_dta("C:/Users/HP Omen/Downloads/census.dta")
```

```
library(ggplot2)

# 5.1 Poblaciones totales de las regiones censales
census %>%
  group_by(region) %>%
  summarise(poblacion = sum(pop))
```

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

```
# 5.2 Poblaciones totales en diagrama de barras
```

```
#5.3 Ordenar estados por población de mayor a menor
census %>%
  arrange(desc(pop))
```

```
## # A tibble: 50 x 12
##       state      region   pop poplt5 pop5_17 pop18p pop65p popurban medage  death
##   <chr>    <dbl+lbl> <dbl> <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
```

```
## 1 Californi~ 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 Pennsylv~ 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>
```

5.4 Nueva variable tasa de divorcios/matrimonios

```
census %>%
  mutate(tasa = divorce/marriage)
```

```
## # A tibble: 50 x 13
##   state      region      pop poplt5 pop5_17 pop18p pop65p popurban medage death
##   <chr>    <dbl+lbl>    <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 Californi~ 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
## 7 Connect~ 1 [NE] 3.11e6 1.85e5 637731 2.28e6 3.65e5 2449774 32 26005
## 8 Delaware 3 [Sout~ 5.94e5 4.12e4 125444 4.28e5 5.92e4 419819 29.8 5123
## 9 Florida 3 [Sout~ 9.75e6 5.70e5 1789412 7.39e6 1.69e6 8212385 34.7 104190
## 10 Georgia 3 [Sout~ 5.46e6 4.15e5 1231195 3.82e6 5.17e5 3409081 28.7 44230
## # ... with 40 more rows, and 3 more variables: marriage <dbl>, divorce <dbl>,
## #   tasa <dbl>
```

5.5 Estados más envejecidos

```
census_mayores = census %>%
  select(state,pop,pop65p,medage) %>%
  mutate(tasa_mayores = pop65p/pop) %>%
  arrange(desc(medage)) %>%
  head(10)
```

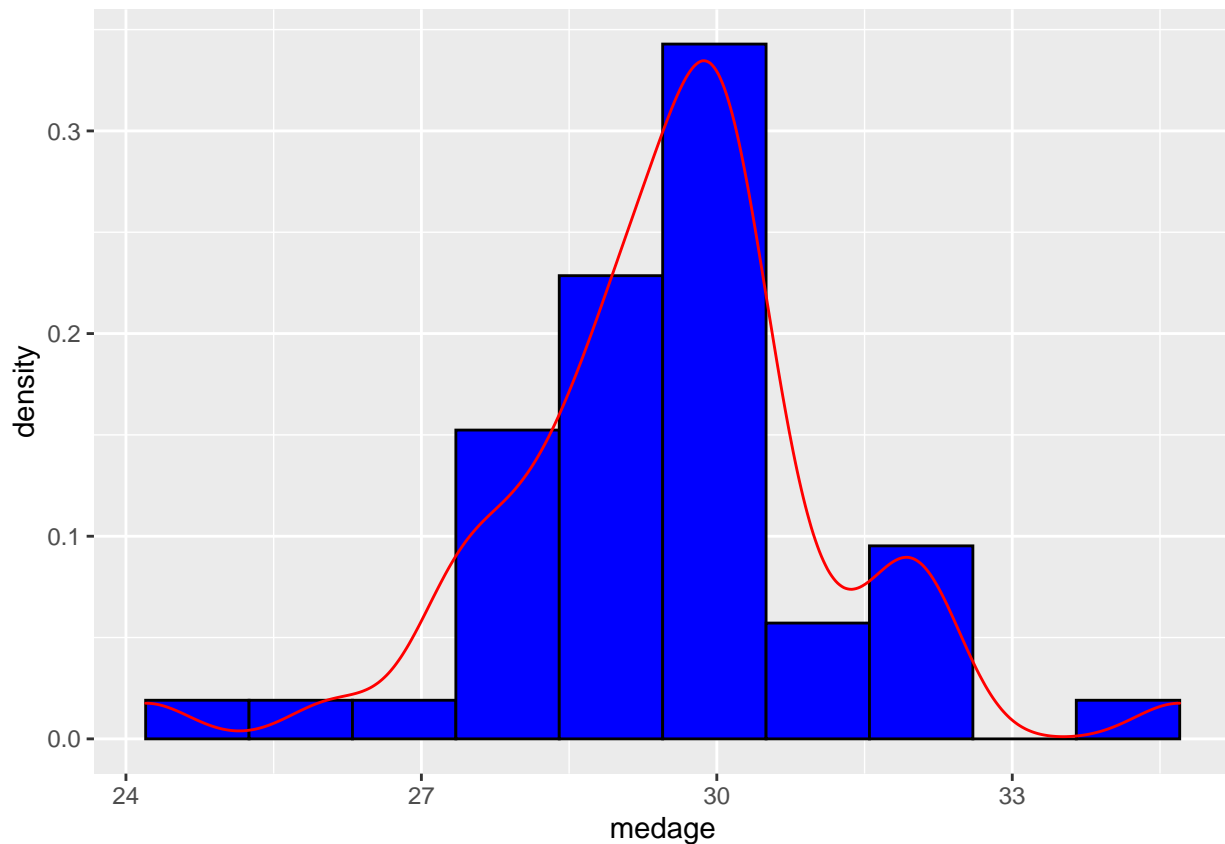
```
census_mayores
```

```
## # A tibble: 10 x 5
##   state      pop pop65p medage tasa_mayores
##   <chr>    <dbl>    <dbl> <dbl>    <dbl>
## 1 Florida 9746324 1687573 34.7      0.173
## 2 New Jersey 7364823 859771 32.2      0.117
## 3 Pennsylvania 11863895 1530933 32.1      0.129
## 4 Connecticut 3107576 364864 32 0.117
## 5 New York 17558072 2160767 31.9      0.123
## 6 Rhode Island 947154 126922 31.8      0.134
## 7 Massachusetts 5737037 726531 31.2      0.127
## 8 Missouri 4916686 648126 30.9      0.132
## 9 Arkansas 2286435 312477 30.6      0.137
## 10 Maine 1124660 140918 30.4      0.125
```

```
# 5.6 Histograma "medage" y curva de densidad
```

```
vec = seq(min(census$medage), max(census$medage), length.out=11)
```

```
ggplot(data=census, aes(x=medage))+ geom_histogram(aes(y=stat(density)), breaks=vec, fill="blue", color="black")
```



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

- Vamos a utilizar el conjunto de datos contenido en el fichero (es un enlace):
cholesterol.csv
Los datos proceden de un estudio realizado en la *University of Virginia School of Medicine* que investiga la prevalencia de la obesidad, la diabetes y otros factores de riesgo cardiovascular. Se puede encontrar más información sobre el fichero en este enlace:
<https://biostat.app.vumc.org/wiki/pub/Main/DataSets/diabetes.html>
- Carga el conjunto de datos en un data.frame de R llamado `chlstr1`.

```
library(tidyverse)
```

```
library(dplyr)
```

```
chlstr1 = read_csv("C:/Users/HP Omen/Documents/MASTER/Fundamentos Matemáticos/datos/cholesterol.csv")
```

```
##
```

```
## -- Column specification -----
```

```
## cols(
##   chol = col_double(),
##   age = col_double(),
##   gender = col_character(),
##   height = col_double(),
##   weight = col_double(),
##   waist = col_double(),
##   hip = col_double()
## )
```

- Empezaremos por información básica sobre el conjunto de datos. Cuántas observaciones contiene, cuáles son las variables y de qué tipos,... Información sobre los datos

```
head(chlstr1)
```

```
## # A tibble: 6 x 7
##   chol age gender height weight waist hip
##   <dbl> <dbl> <chr>   <dbl>   <dbl> <dbl> <dbl>
## 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
```

```
#número de filas
nrow(chlstr1)
```

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

```
#número de columnas
ncol(chlstr1)
```

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

```
#nombres de las variables
names(chlstr1)
```

```
## [1] "chol" "age" "gender" "height" "weight" "waist" "hip"
```

```
#clase de las variables
chlstr1 %>%
  sapply(class)
```

```
##      chol      age      gender      height      weight      waist
## "numeric" "numeric" "character" "numeric" "numeric" "numeric"
##      hip
## "numeric"
```

- Asegúrate de comprobar si hay datos ausentes y localízalos en la tabla. NA's

```
colSums(is.na(chlstr1))
```

```
## chol age gender height weight waist hip
## 1 0 0 5 1 2 2
```

- 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. El análisis debe contener, al menos:
 - Para las variables cuantitativas (continuas o discretas).
Resumen numérico básico.
Gráficas (las adecuadas, a ser posible más de un tipo de gráfico).
 - Variables categóricas (factores).
Tablas de frecuencia (absolutas y relativas).
Gráficas (diagrama de barras).

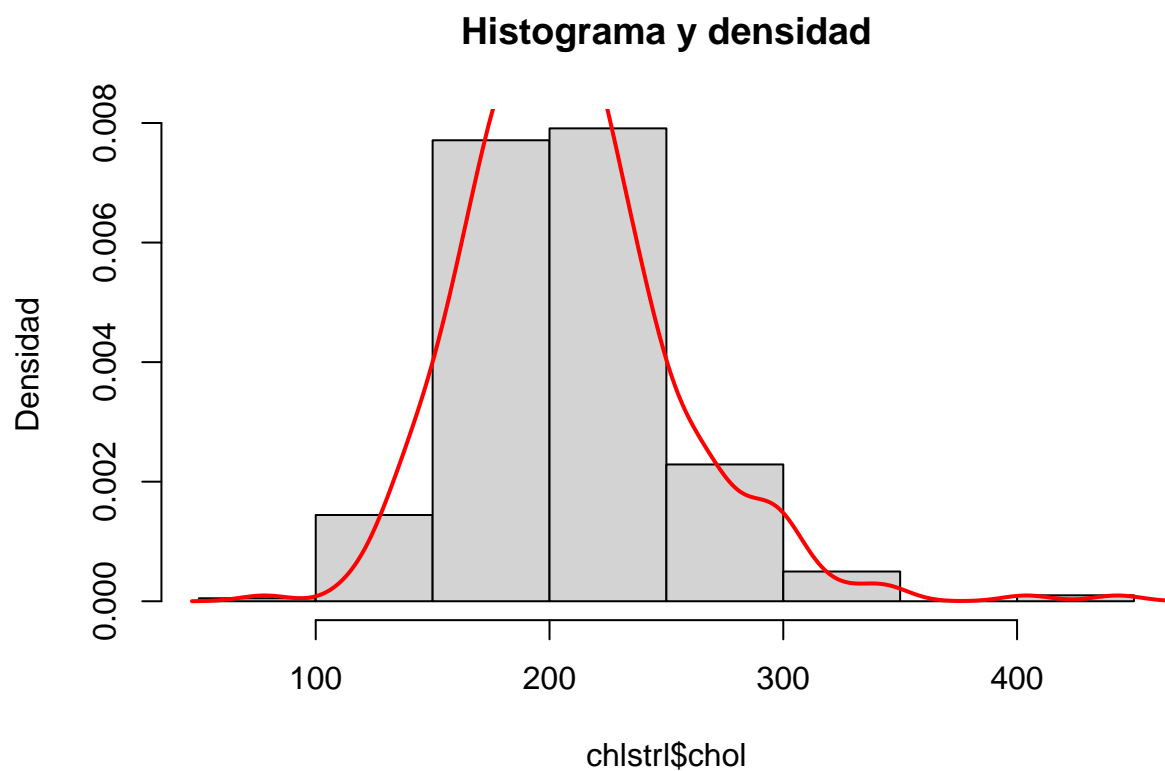
Análisis Exploratorio de los datos

```
summary(chlstr1)
```

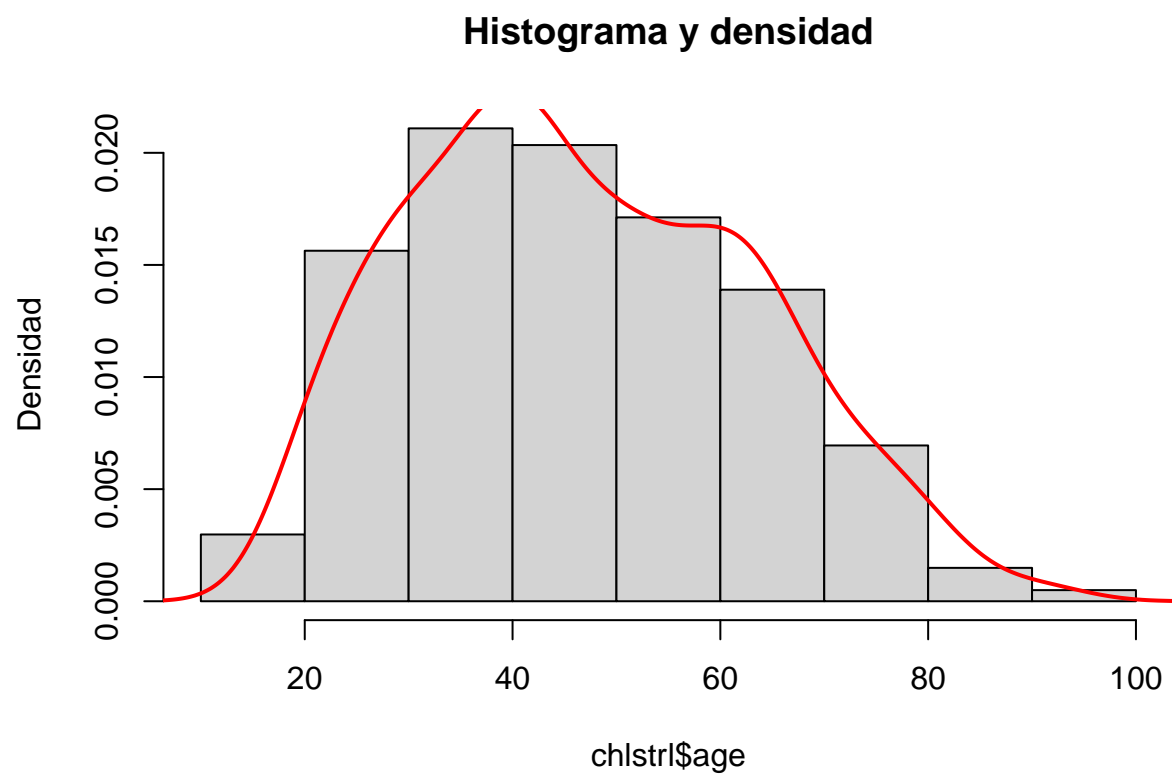
```
## chol age gender height
## Min. : 78.0 Min. :19.00 Length:403 Min. :52.00
## 1st Qu.:179.0 1st Qu.:34.00 Class :character 1st Qu.:63.00
## Median :204.0 Median :45.00 Mode :character Median :66.00
## Mean :207.8 Mean :46.85 Mean :66.02
## 3rd Qu.:230.0 3rd Qu.:60.00 3rd Qu.:69.00
## Max. :443.0 Max. :92.00 Max. :76.00
## NA's :1 NA's :5
## weight waist hip
## Min. : 99.0 Min. :26.0 Min. :30.00
## 1st Qu.:151.0 1st Qu.:33.0 1st Qu.:39.00
## Median :172.5 Median :37.0 Median :42.00
## Mean :177.6 Mean :37.9 Mean :43.04
## 3rd Qu.:200.0 3rd Qu.:41.0 3rd Qu.:46.00
## Max. :325.0 Max. :56.0 Max. :64.00
## NA's :1 NA's :2 NA's :2
```

```
#Variables Numéricas
```

```
hist(chlstr1$chol,freq = FALSE, main = "Histograma y densidad",
     ylab = "Densidad")
dx <- density(chlstr1$chol,na.rm = TRUE)
lines(dx, lwd = 2, col = "red")
```

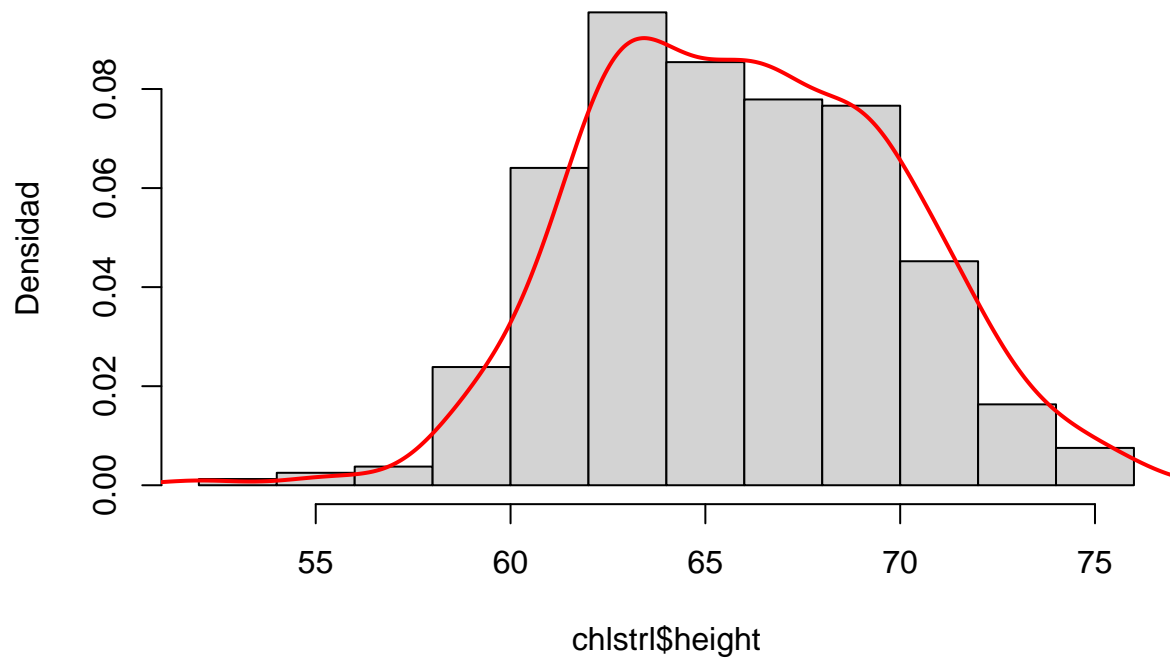


```
hist(chlst1$age,freq = FALSE, main = "Histograma y densidad",  
      ylab = "Densidad")  
dx <- density(chlst1$age,na.rm = TRUE)  
lines(dx, lwd = 2, col = "red")
```

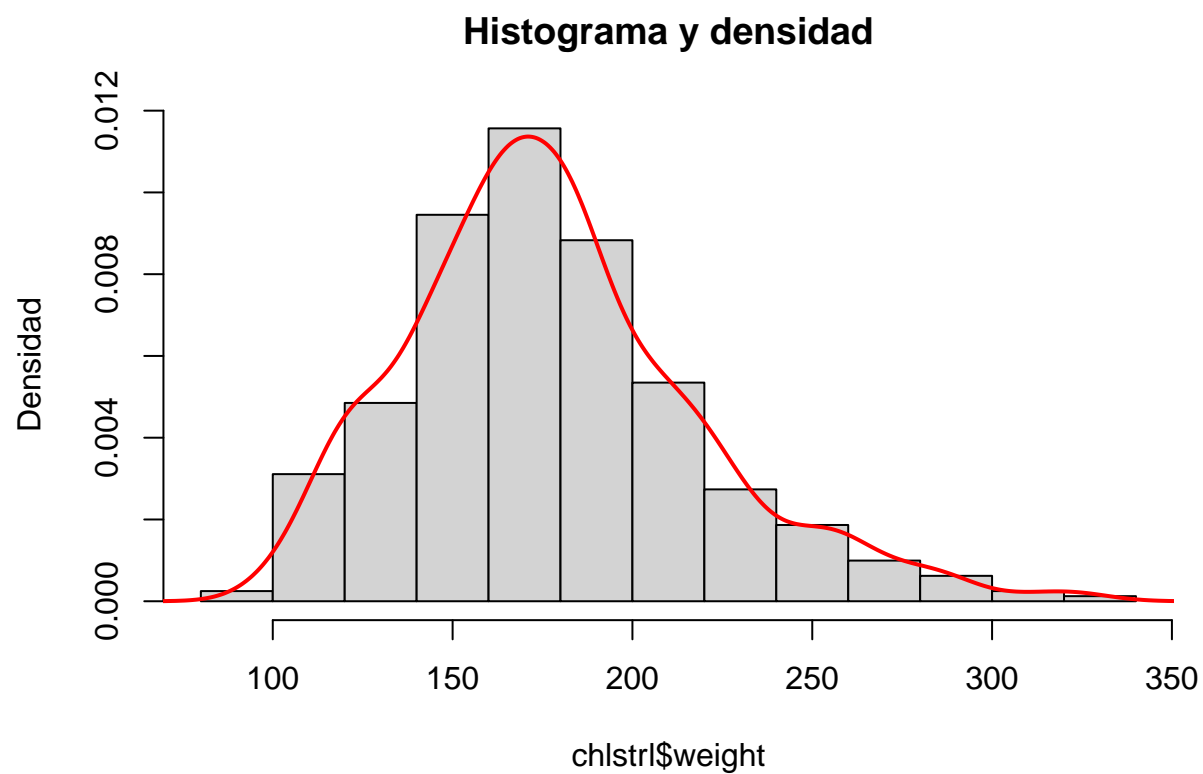



```
hist(chlst1$height,freq = FALSE, main = "Histograma y densidad",  
      ylab = "Densidad")  
dx <- density(chlst1$height,na.rm = TRUE)  
lines(dx, lwd = 2, col = "red")
```

Histograma y densidad

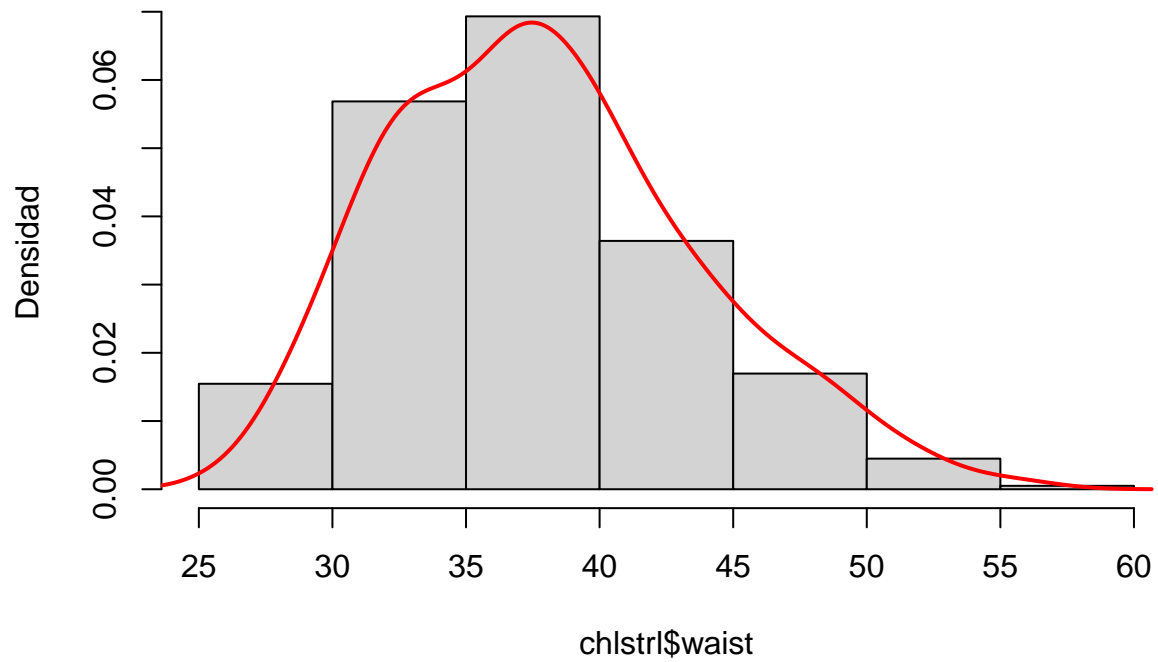


```
hist(chlstrl$weight,freq = FALSE, main = "Histograma y densidad",  
      ylab = "Densidad")  
dx <- density(chlstrl$weight,na.rm = TRUE)  
lines(dx, lwd = 2, col = "red")
```



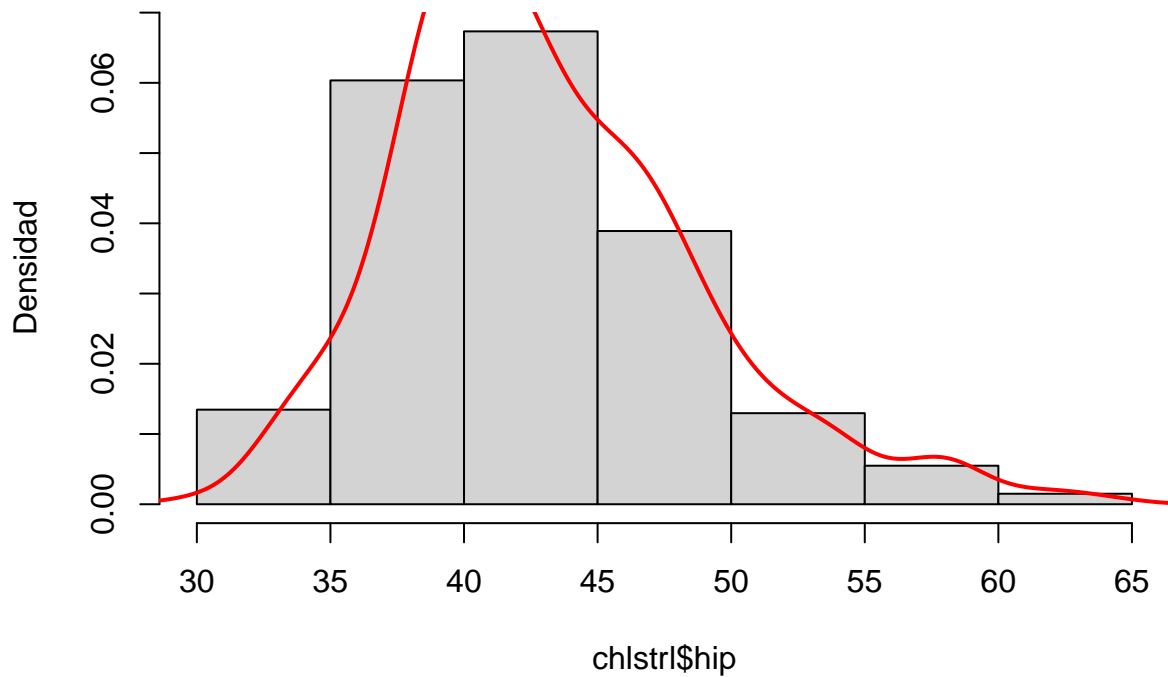
```
hist(chlst1$waist,freq = FALSE, main = "Histograma y densidad",  
      ylab = "Densidad")  
dx <- density(chlst1$waist,na.rm = TRUE)  
lines(dx, lwd = 2, col = "red")
```

Histograma y densidad



```
hist(chlst1$hip,freq = FALSE, main = "Histograma y densidad",  
      ylab = "Densidad")  
dx <- density(chlst1$hip,na.rm = TRUE)  
lines(dx, lwd = 2, col = "red")
```

Histograma y densidad



#Variables Categóricas

```
table(chlstrl$gender)
```

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

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

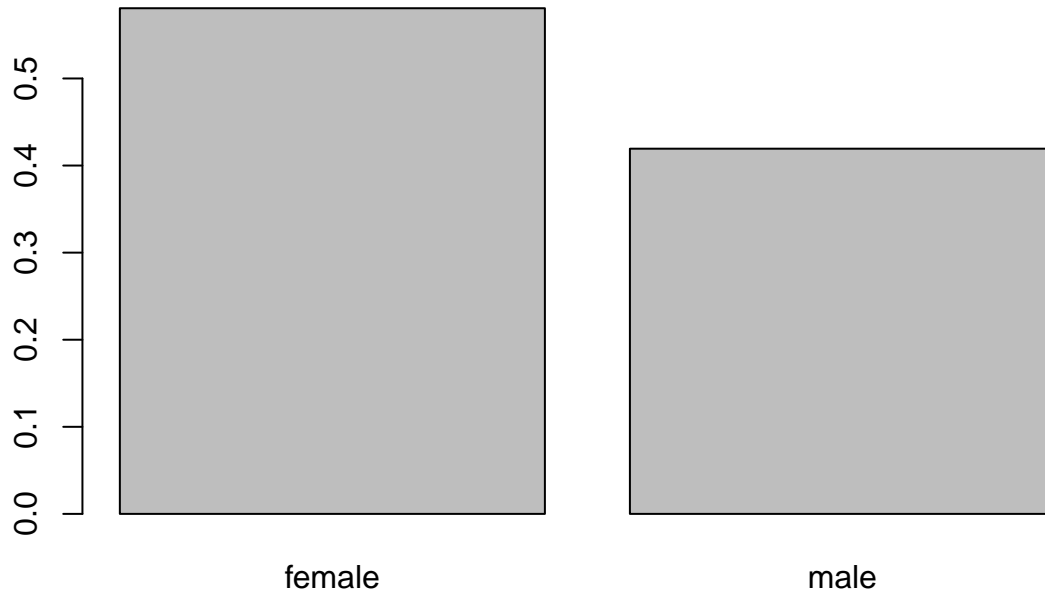
```
##
##   female      male
## 0.5806452 0.4193548
```

#medias de variables numéricas según género

```
chlstrl %>%
  group_by(gender) %>%
  summarise(meanChol = mean(chol, na.rm = TRUE), meanAge = mean(age, na.rm = TRUE), meanHeight = mean(height, na.rm = TRUE), meanWeight = mean(weight, na.rm = TRUE), meanWaist = mean(waist, na.rm = TRUE), meanHip = mean(hip, na.rm = TRUE))
```

```
## # A tibble: 2 x 7
##   gender meanChol meanAge meanHeight meanWeight meanWaist meanHip
##   <chr>    <dbl>   <dbl>    <dbl>    <dbl>    <dbl>   <dbl>
## 1 female    208.    45.8     63.7     174.    38.1    44.3
## 2 male      207.    48.3     69.1     182.    37.6    41.2
```

```
barplot(prop.table(table(chlstr1$gender)))
```



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

```
library(dplyr)
chlstr10 = chlstr1 %>%
  mutate(weight = weight * 0.454, height = height * 0.0254)
chlstr10
```

```
## # A tibble: 403 x 7
##   chol  age gender height weight waist  hip
##   <dbl> <dbl> <chr>   <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
## 7  195   30 male     1.75    86.7   46    49
## 8  227   37 male     1.50    77.2   34    39
## 9  177   45 male     1.75    75.4   34    40
## 10 263   55 female   1.60    91.7   45    50
## # ... with 393 more rows
```

- Ahora usa esos valores de `height` y `weight` para añadir una nueva columna llamada BMI, definida mediante:

$$BMI = \frac{weight}{height^2}$$

(se divide por el cuadrado de la altura).

```
chlstr10 = chlstr10 %>%
  mutate(BMI = weight/(height^2))
chlstr10
```

```
## # A tibble: 403 x 8
##   chol   age gender height weight waist   hip   BMI
##   <dbl> <dbl> <chr>   <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
## 7   195    30 male     1.75   86.7    46    49   28.2
## 8   227    37 male     1.50   77.2    34    39   34.4
## 9   177    45 male     1.75   75.4    34    40   24.5
## 10  263    55 female   1.60   91.7    45    50   35.8
## # ... with 393 more rows
```

- Crea una nueva columna llamada `ageGroup` dividiendo la edad en los siguientes tres niveles:

(10,40], (40,70], (70,100]

```
chlstr10 = chlstr10 %>%
  mutate(ageGroup = cut(age,breaks = c(10,40,70,100)))
chlstr10
```

```
## # A tibble: 403 x 9
##   chol   age gender height weight waist   hip   BMI ageGroup
##   <dbl> <dbl> <chr>   <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]
## 7   195    30 male     1.75   86.7    46    49   28.2 (10,40]
## 8   227    37 male     1.50   77.2    34    39   34.4 (10,40]
## 9   177    45 male     1.75   75.4    34    40   24.5 (40,70]
## 10  263    55 female   1.60   91.7    45    50   35.8 (40,70]
## # ... with 393 more rows
```

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

```
chlstrl0 %>%
  group_by(ageGroup) %>%
  summarise(n = n())
```

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

```
chlstrl0 %>%
  filter(gender == "female") %>%
  group_by(ageGroup) %>%
  summarise(meanChol = mean(chol, na.rm = TRUE),
            meanBMI = mean(BMI, na.rm = TRUE))
```

```
## # A tibble: 3 x 3
##   ageGroup meanChol meanBMI
##   <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).

```
#ARREGLAR

y = c(-12,-19,9,-13,-14,-17,8,-19,-14)
u = c(1,-1,3)
cambiosSigno = function(x){
  difs = 0
  for(i in 1:(length(x)-1)){

    if(sign(x[i+1])!=sign(x[i])){
      difs = difs+1
    }
  }
  print(difs)
}

vec = sample(c(-100:1, 1:100), 20, replace = TRUE)
cambiosSigno(vec)
```



```
## [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`

```
#METER SAMPLE
```

```
cambiosSignoPos = function(x){  
  posiciones = c()  
  difs = 0  
  for(i in 1:(length(x)-1)){  
    if(sign(x[i+1])!=sign(x[i])){  
      difs = difs+1  
      posiciones = c(posiciones,i+1)  
    }  
  }  
  print(difs)  
  print(posiciones)  
}  
cambiosSignoPos(y)
```

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

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

También se valorará que incluyas en el código como usar ‘sample’ para generar vectores aleatorios de 20

Ejercicio 3. R4DS.

Es recomendable que esta semana del curso hagas al menos una lectura somera de los Capítulos 1 a 5 de R for Data Science (R4DS), de H. Wickham, con énfasis especial en los Capítulos 3 y 5 (los capítulos 1, 2 y 4 son muy breves). Los siguientes apartados pretenden motivar esa lectura y por eso mismo pueden resultar un poco más laboriosos.

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

Gráfico 1:

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

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

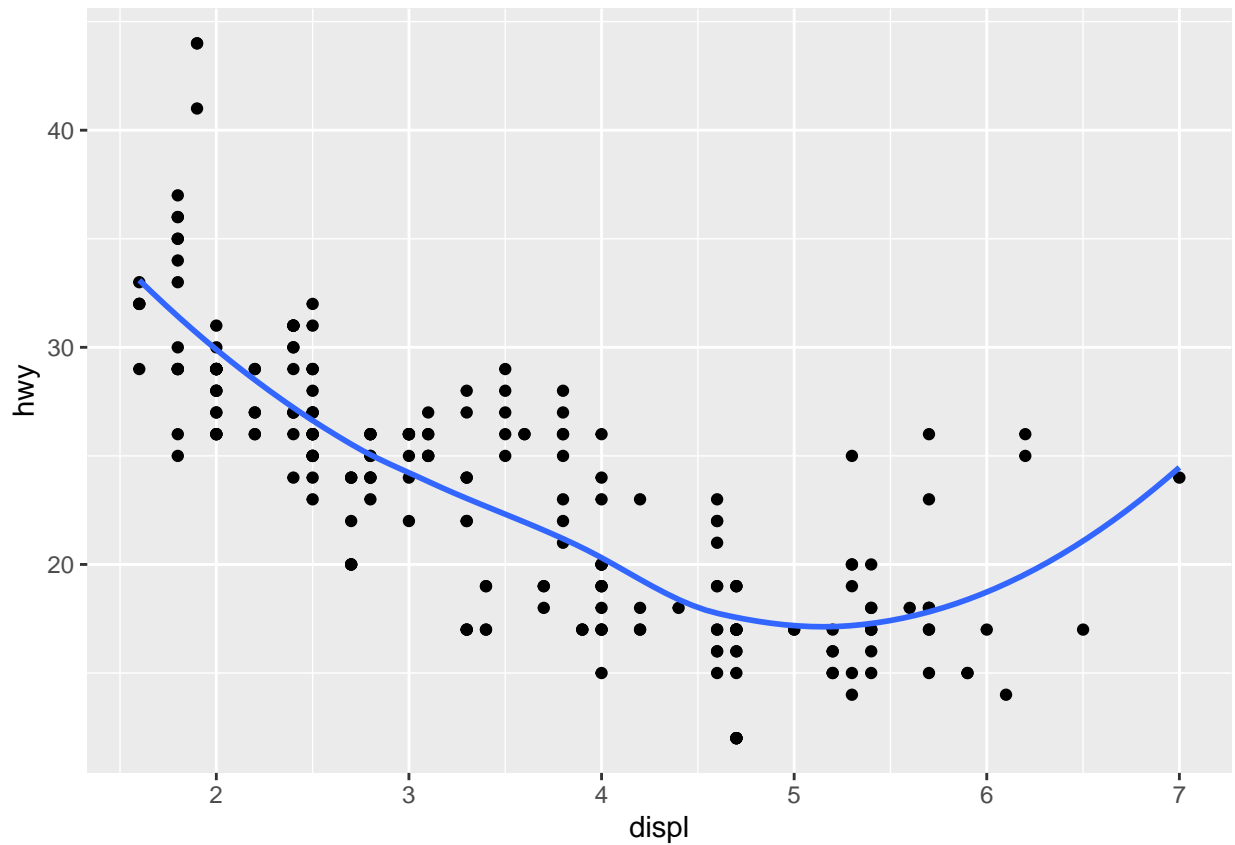


Gráfico 2:

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

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

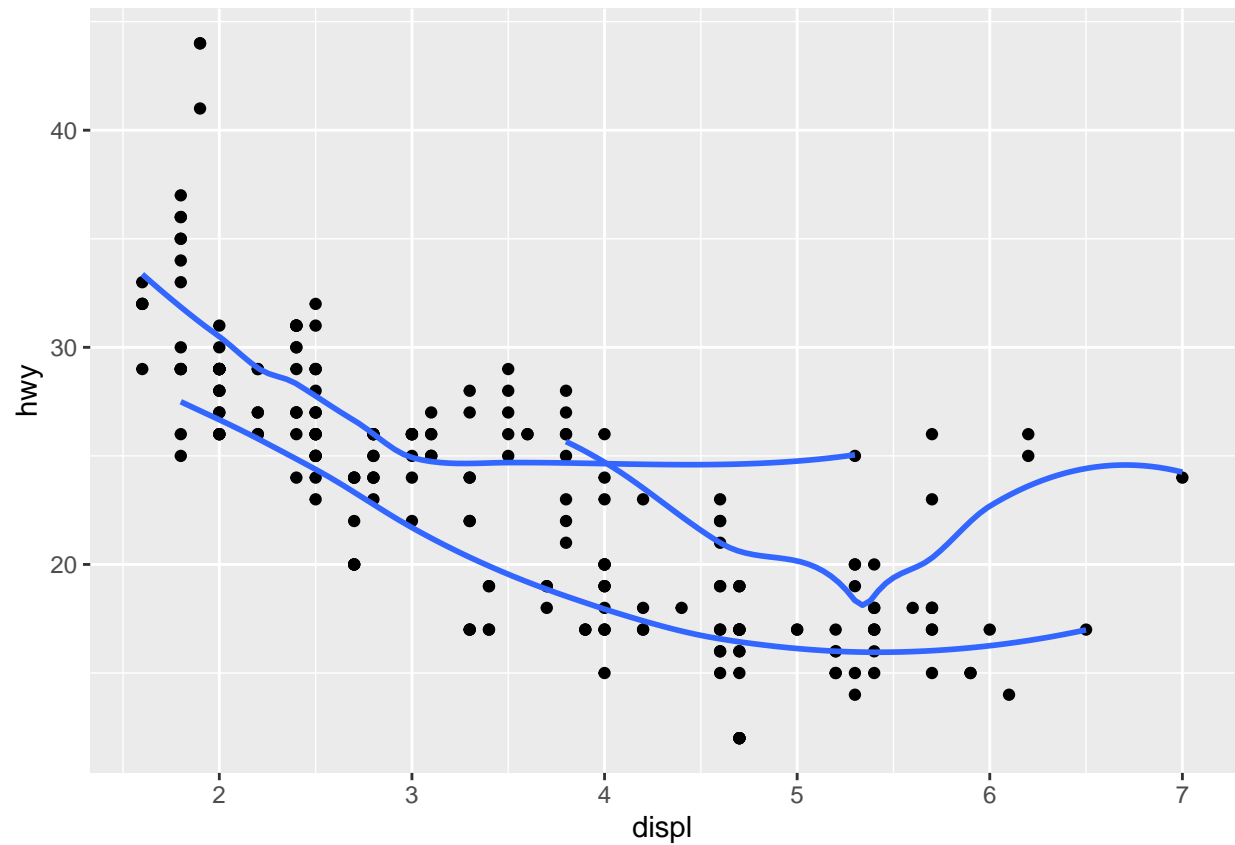


Gráfico 3:

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

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

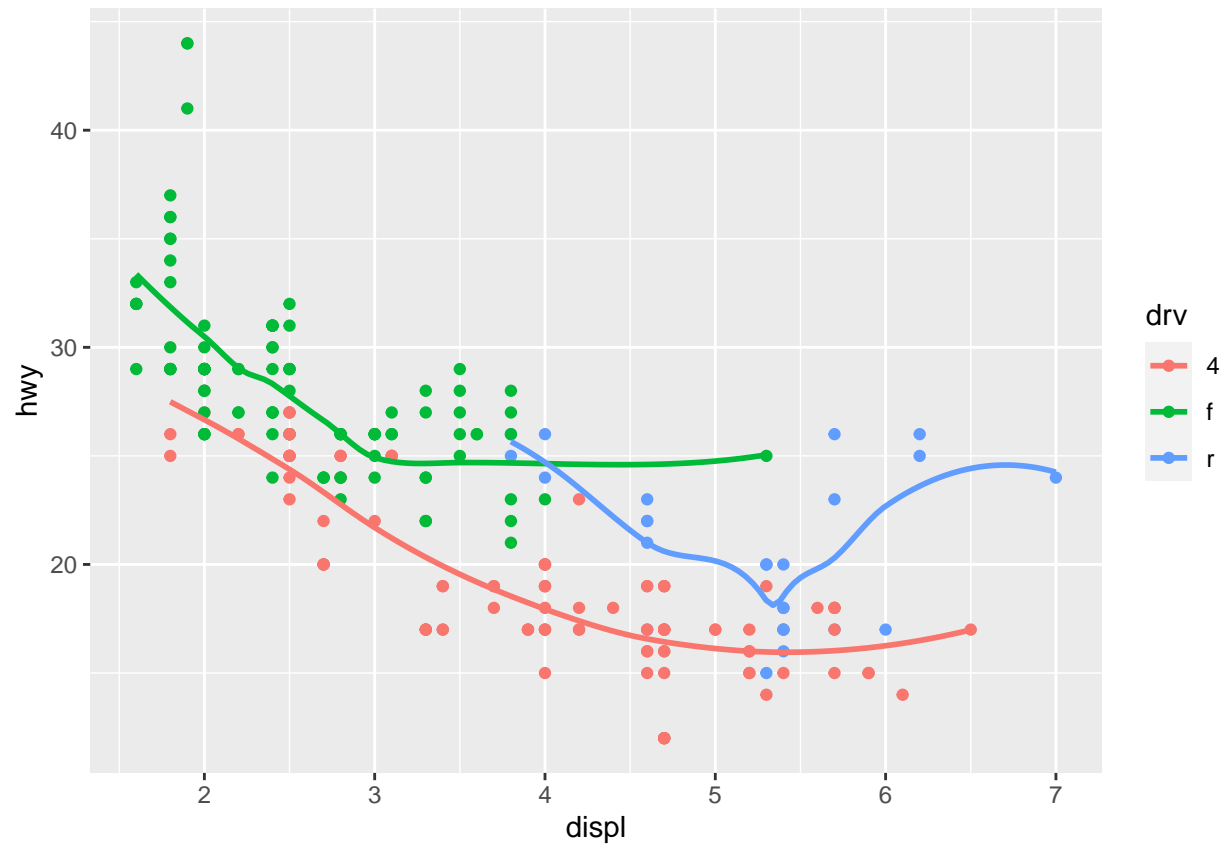


Gráfico 4:

```
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)
```

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

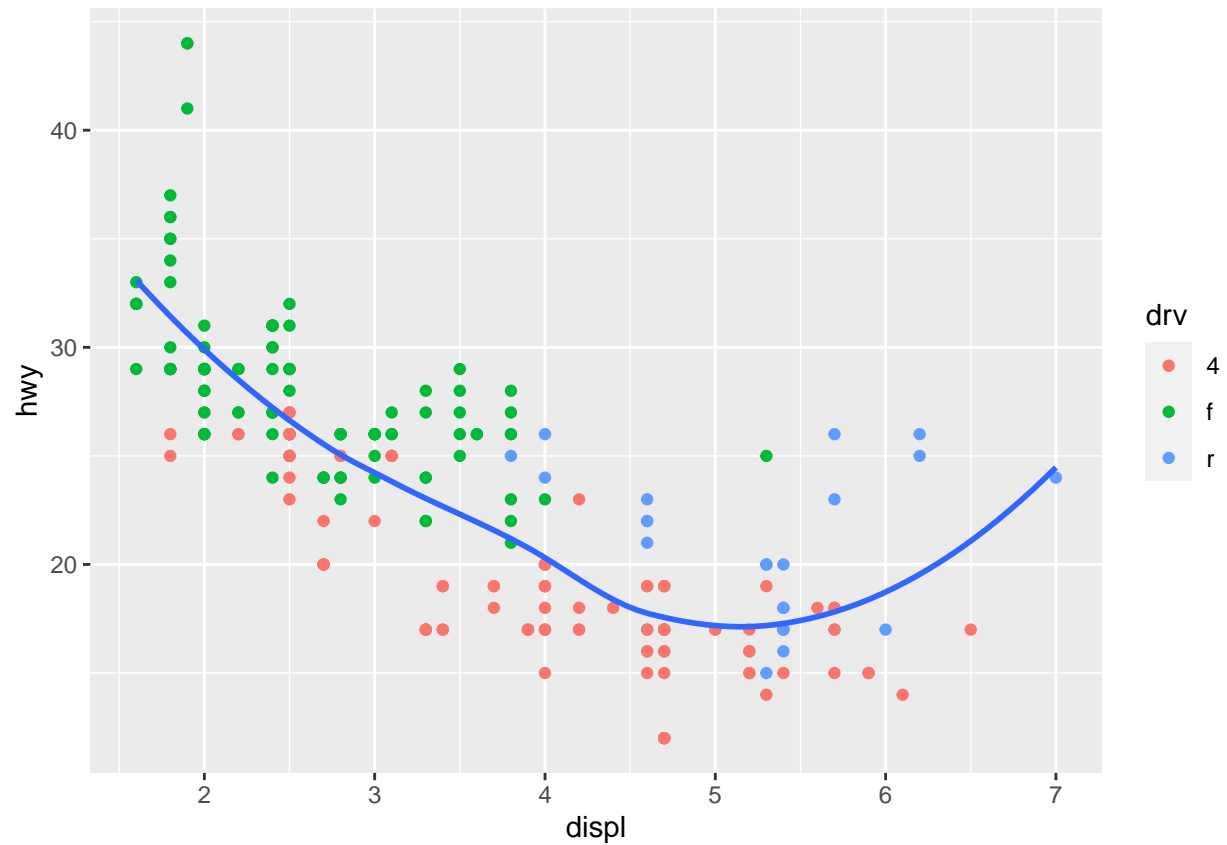


Gráfico 5:

```
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)
```

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

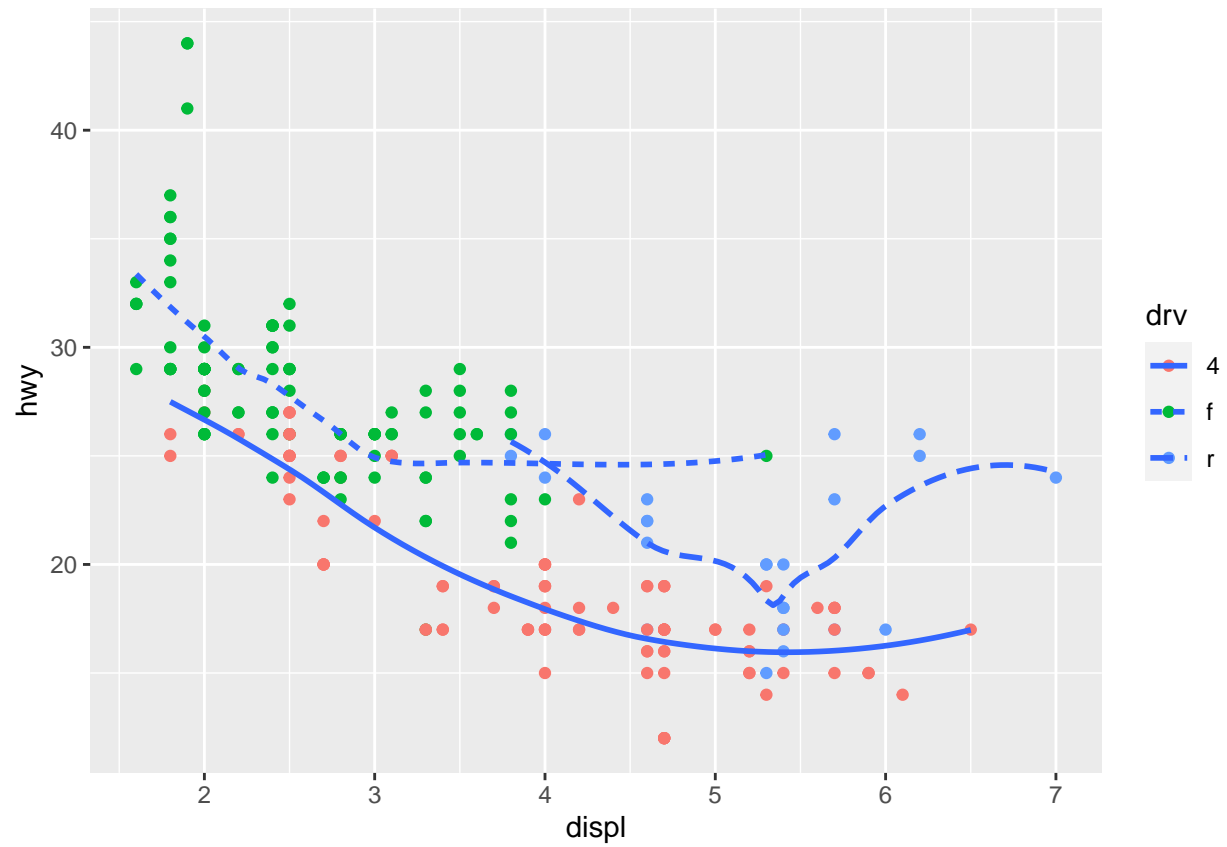
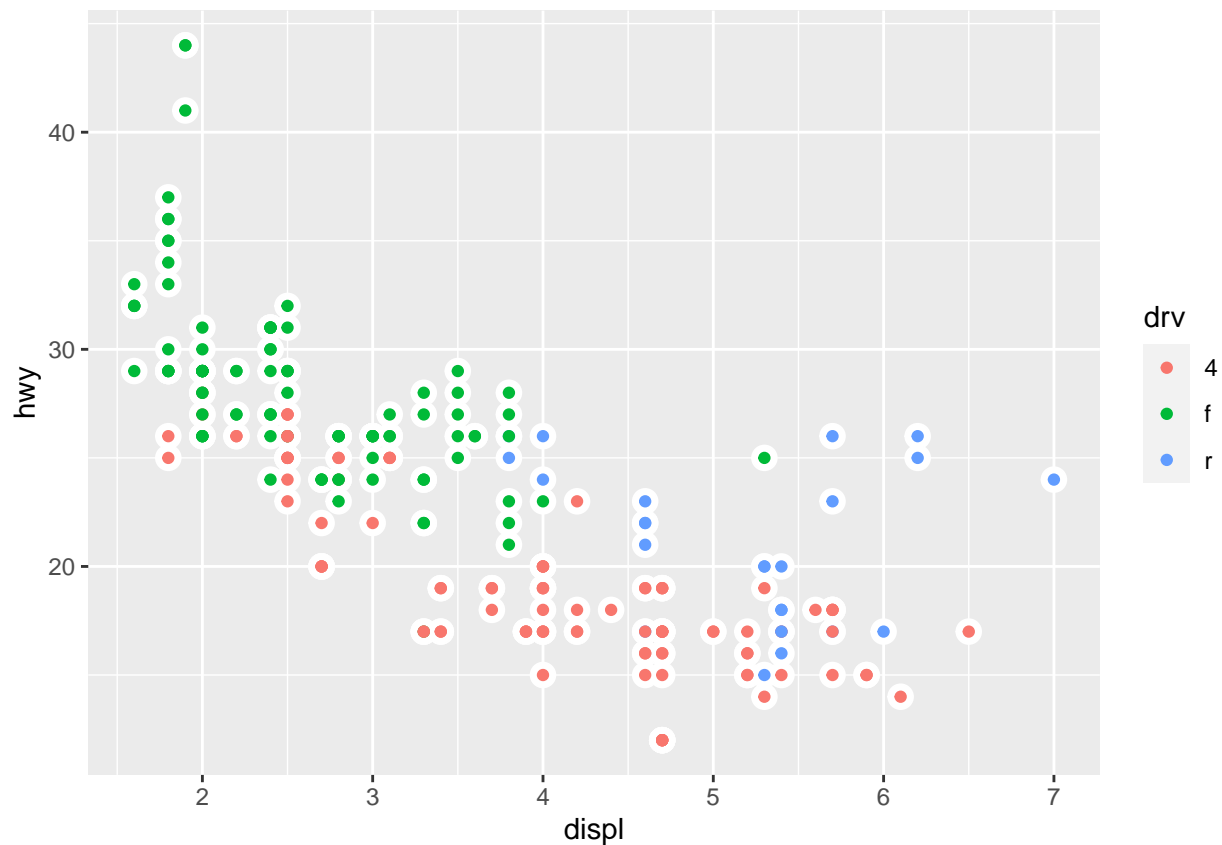


Gráfico 6:

```
ggplot(mpg, aes(x = displ, y = hwy)) +  
  geom_point(size = 4, color = "white") +  
  geom_point(aes(colour = drv))
```



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

```
library(nycflights13)
```

```
## Warning: package 'nycflights13' was built under R version 4.1.1
```

```
flights
```

```
## # A tibble: 336,776 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 336,766 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 <dtm>
```

```
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 <dtm>
```

```
flights %>%
  filter(dest == "IAH"|dest == "HOU")
```

```
## # 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 <dtm>
```

```
flights %>%
  filter(carrier == "DL"|carrier == "UA"|carrier == "")
```

```
## # A tibble: 106,775 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     554             600         -6     812             837
## 4  2013     1     1     554             558         -4     740             728
## 5  2013     1     1     558             600         -2     924             917
## 6  2013     1     1     558             600         -2     923             937
```



```
## 7 2013 1 1 559 600 -1 854 902
## 8 2013 1 1 602 610 -8 812 820
## 9 2013 1 1 606 610 -4 837 845
## 10 2013 1 1 607 607 0 858 915
## # ... with 106,765 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 <dtm>
```

```
flights %>%
  filter(month == 7|month == 8| month == 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 <dtm>
```

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

```
## # A tibble: 3 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    10     7   1350         1350         0    1736         1526
## 2 2013     5    23   1810         1810         0    2208         2000
## 3 2013     7     1    905         905         0    1443         1223
## # ... with 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 <dtm>
```

```
#MAL
flights %>%
  filter(dep_delay >= 60 & arr_delay <= 30)
```

```
## # A tibble: 239 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     3   1850         1745         65    2148         2120
## 2 2013     1     3   1950         1845         65    2228         2227
## 3 2013     1     3   2015         1915         60    2135         2111
## 4 2013     1     6   1019         900         79    1558         1530
```

```
## 5 2013 1 7 1543 1430 73 1758 1735
## 6 2013 1 11 1020 920 60 1311 1245
## 7 2013 1 12 1706 1600 66 1949 1927
## 8 2013 1 12 1953 1845 68 2154 2137
## 9 2013 1 19 1456 1355 61 1636 1615
## 10 2013 1 21 1531 1430 61 1843 1815
## # ... with 229 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 <dtm>
```

```
flights %>%
  filter(dep_time >= 0000 & dep_time <= 0600)
```

```
## # A tibble: 9,344 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 9,334 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 <dtm>
```