Master en Big Data. Fundamentos Matemáticos del Análisis de Datos (FMAD).

Tarea 1

Gutiérrez García, Laura

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

# Librerías

Antes de comenzar con la práctica, cargamos todas las librerías necesarias:

library(tidyverse) # Uso de dplyr y ggplot  
library(gridExtra) # Mostrar varios gráficos juntos  
library(nycflights13) # Base de datos Ejercicio 3

# 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.

# 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](https://gist.githubusercontent.com/fsansegundo/ee991e53e1a571dd34034c42b5516eae/raw/2206455b5772e90c5a2a24a3f42a84408fd1d1c5/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 chlstrl.

Leemos el conjunto de datos con la función read\_csv de la librería tidyverse y mostramos el contenido de las 6 primeras filas:

chlstrl <- read\_csv("./data/cholesterol.csv")  
head(chlstrl)

## # 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

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

Número de observaciones y variables:

nrow(chlstrl) # Nº filas = nº observaciones

## [1] 403

ncol(chlstrl) # Nº columnas = nº variables

## [1] 7

dim(chlstrl) # Nº filas x nº columnas

## [1] 403 7

Cuáles son las variables:

names(chlstrl)

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

str(chlstrl)

## spec\_tbl\_df [403 x 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>

Con la función “names” vemos el nombre de las variables del conjunto de datos y con la función “str” se indica el tipo de dato que almacenan dichas variables. En nuestro caso, todas son numéricas salvo el sexo que es de tipo carácter.

* Asegúrate de comprobar si hay datos ausentes y localízalos en la tabla.

head(is.na(chlstrl))

## chol age gender height weight waist hip  
## [1,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [2,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [3,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [4,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [5,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
## [6,] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

head(complete.cases(chlstrl))

## [1] TRUE TRUE TRUE TRUE TRUE TRUE

summary(chlstrl)

## 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

Con la función “is.na” devuelve un dataframe con los valores lógicos True/False indicando dónde se encuentran los datos faltantes. Por otra parte, la función “complete.cases” nos indica si en la fila hay algún dato faltante o no (True si la fila está completa y False si hay algún missing) y la función summary, además de informarnos sobre los descriptivos de cada variable (mínimo, máximo, cuartiles y media) también nos indica el número de dadtos faltantes en cada una de ellas.

Si quisiéramos seleccionar las filas donde no hay ningún dato faltante:

chlstrl[complete.cases(chlstrl),]

## # A tibble: 394 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  
## 7 195 30 male 69 191 46 49  
## 8 227 37 male 59 170 34 39  
## 9 177 45 male 69 166 34 40  
## 10 263 55 female 63 202 45 50  
## # ... with 384 more rows

Sin embargo, a pesar del escaso número de estos datos faltantes, como depende de cada variable de estudio, se tratarán de acuerdo a la característica que se esté analizando incluyendo como argumento na.rm = T en las funciones de los ejercicios posteriores.

* 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 para la variable cuantitativa colesterol (“chol”):

summary(chlstrl$chol)

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

# Recorrido intercuartílico  
IQR(chlstrl$chol, na.rm = T)

## [1] 51

# Valores atípicos  
unname(quantile(chlstrl$chol, probs = c(1/4, 3/4), na.rm = T) + c(-1, 1) \* 1.5 \* IQR(chlstrl$chol, na.rm = T))

## [1] 102.5 306.5

Mediante el resumen de esta variable, se observa que los valores para el colesterol oscilan entre un mínimo de 78 y un máximo de 443 obteniendo un nivel medio de 207.8. El primer cuartil nos indica que el 25% de los datos se encuentra por debajo de un nivel de 179, la mediana que el 50% de los pacientes ha tenido unos niveles superiores a 204 y el tercer cuartil que el 75% de los pacientes tiene un nivel inferior a 230. Por otro lado, en esta variable solo hay un dato ausente (NA).

Continuando con el recorrido intercuartílico, con él se puede estudiar la dispersión (a partir de la diferencia entre el tercer y el primer cuartil), de tal forma que, cuánto mayor sea su valor, mayor es la dispersión.

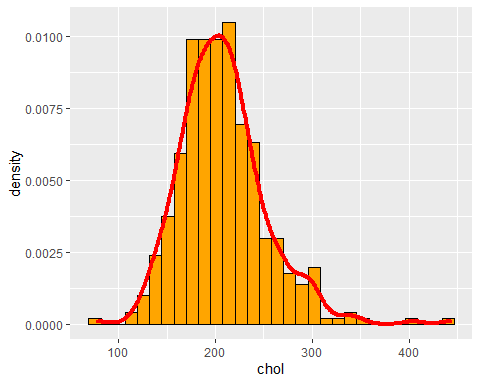
En cuanto a los valores atípicos,se definen como aquellos que estarán fuera del rango (102.5,306.5).

A continuación, se realiza el histograma con la curva de densidad:

ggplot(chlstrl, aes(x = chol)) +   
 geom\_histogram(aes(y=stat(density)),   
 fill = "orange", color="black") +   
 geom\_density(color="red", size=1.5)

## Warning: Removed 1 rows containing non-finite values (stat\_bin).

## Warning: Removed 1 rows containing non-finite values (stat\_density).

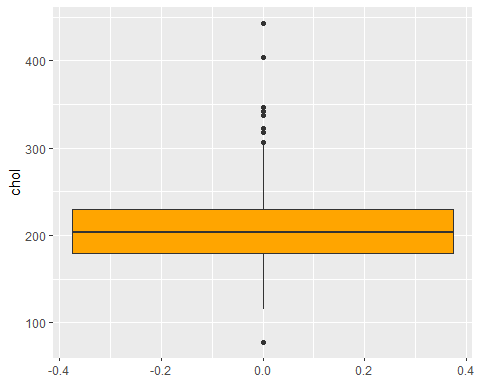


En el histograma se aprecia como el nivel de colesterol se asemeja a una distribución Normal con una cola alargada hacia la derecha (asimétrica por la derecha).

Ahora, vamos a representar los valores mediante un diagrama de cajas (boxplot) y un violinplot:

# Boxplot  
ggplot(chlstrl) +  
geom\_boxplot(mapping = aes(y = chol), fill="orange")

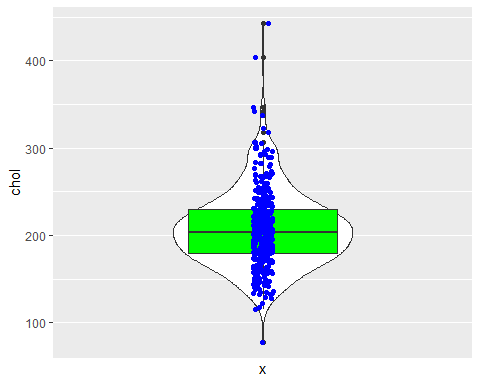
## Warning: Removed 1 rows containing non-finite values (stat\_boxplot).



# Violinplot  
ggplot(chlstrl) +  
geom\_violin(mapping = aes(x=0, y = chol)) +  
scale\_x\_discrete(breaks = c()) +  
geom\_boxplot(mapping = aes(y = chol), fill="green") +  
geom\_jitter(aes(x=0, y = chol),  
position = position\_jitter(w=0.05, h= 0), col="blue")

## Warning: Removed 1 rows containing non-finite values (stat\_ydensity).  
  
## Warning: Removed 1 rows containing non-finite values (stat\_boxplot).

## Warning: Removed 1 rows containing missing values (geom\_point).



Como ya se describió en el resumen numérico, la media de los valores se encuentra en torno a los 200 y hay ciertos outliers que sobresalen del recorrido intercuartílico convirtiéndose en valores extremos (por debajo de 100 y encima de 300).

Otras medidas de dispersión son:

* Desviación absoluta mediana

mad(chlstrl$chol, na.rm = TRUE)

## [1] 37.065

* Varianza y desviación típica muestrales

var(chlstrl$chol, na.rm = TRUE) # Varianza

## [1] 1975.408

sd(chlstrl$chol, na.rm = TRUE) # Desviación típica

## [1] 44.44556

Análisis exploratorio para la variable categórica : sexo (“gender”)

Tablas de frecuencia (absolutas y relativas).

* Frecuencias absolutas:

chlstrl %>%   
 count(gender)

## # A tibble: 2 x 2  
## gender n  
## <chr> <int>  
## 1 female 234  
## 2 male 169

* Frecuencias relativas:

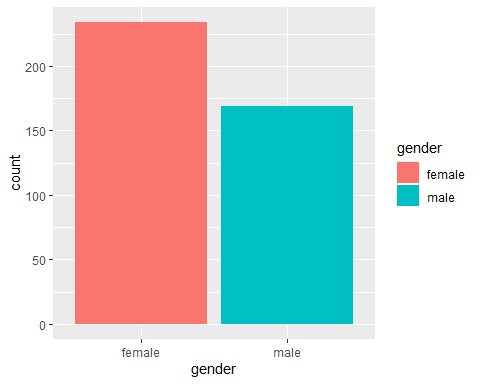
chlstrl %>%   
 count(gender) %>%  
 mutate(gender, relFreq = prop.table(n), n=NULL)

## # A tibble: 2 x 2  
## gender relFreq  
## <chr> <dbl>  
## 1 female 0.581  
## 2 male 0.419

El 58% de los pacientes son mujeres y el 42% hombres.

* Diagrama de barras con la frecuencia absoluta:

ggplot(chlstrl) +  
geom\_bar(mapping = aes(x = gender, fill=gender))



* Los valores de height y weight están en pulgadas (inches) y libras (pounds) respectivamente. Una libra 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.

Hago el cambio con las variables altura y peso y lo guardo en otro dataframe para no sobrescribir los cambios en la base de datos original:

chlstrl2 <- chlstrl %>%   
 mutate(height = height\*0.0254,  
 weight = weight\*0.454)

* Ahora usa esos valores de height y weight para añadir una nueva columna llamada BMI, definida mediante:
* (se divide por el cuadrado de la altura).

Se añade la nueva columna denomida como BMI al dataframe y mostramos las 6 primeras filas del conjunto de datos resultante:

chlstrl2 <- chlstrl2 %>%   
 mutate(BMI = weight/height^2)  
  
head(chlstrl2)

## # A tibble: 6 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

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

chlstrl2 <- chlstrl2 %>%   
 mutate(ageGroup = cut(age, breaks = c(10, 40, 70, 100) ))  
  
head(chlstrl2)

## # A tibble: 6 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]

* Usando dplyr calcula cuántas observaciones hay en cada nivel de ageGroup (indicación: usa group\_by).

chlstrl2 %>%   
 group\_by(ageGroup) %>%   
 count(ageGroup)

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

El grupo de 40 a 70 años es el que más observaciones tiene (207).

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?

Filtrando por las filas de sexo femenino y agrupando por el grupo de edad, se calculan las medias para las variables “chol” y “BMI”:

chlstrl2 %>%   
 filter(gender=="female") %>%   
 group\_by(ageGroup) %>%   
 summarise(chol.mean = mean(chol, na.rm = T),  
 BMI.mean = mean(BMI, na.rm = T))

## # A tibble: 3 x 3  
## ageGroup chol.mean BMI.mean  
## <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).

Para construir la función, primeramente se crea el vector lógico signo que almacena True si el número es positivo y False si es negativo. Como los vectores lógicos se tratan en R como 1 y 0, se le puede aplicar la función diff que calcula la diferencia entre las posiciones consecutivas de un vector. Así, si hacemos el valor absoluto de este resultado y lo sumamos, obtendremos el número de cambios de signo que ha habido.

Por otro lado, a la función se le incluye como argumento la generación de un vector aleatorio para que, en caso de no introducir ningún vector, se genere uno por defecto. Como resultado, la función devuelve una lista con el vector introducido (o generado) y el número de cambios de signo:

cambiosSigno <- function(x=sample(c(-1, 1), 9, replace = TRUE) \* sample(1:20, 9, replace = TRUE)){  
 sol <- list()  
 sol$vector <- x  
 signo <- x>0  
 sol$cambios <- sum(abs(diff(signo)))  
 return(sol)  
}

* 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

Al igual que antes, la función, devuelve una lista con el vector introducido (o generado) y, en este caso, las posiciones donde se ha producido el cambio de signo:

cambiosSignoPos <- function(x = sample(c(-1, 1), 9, replace = TRUE) \* sample(1:20, 9, replace = TRUE)){  
 sol <- list()  
 sol$vector <- x  
 signo <- x>0  
 cambio <- c(0,abs(diff(signo)))  
 sol$pos <- which(cambio==1)  
 return(sol)  
}

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).

Para comprobar el funcionamiento de las funciones se emplea el vector del ejemplo y se llama a la función sin argumento:

set.seed(2019)  
(x <- sample(c(-1, 1), 9, replace = TRUE) \* sample(1:20, 9, replace = TRUE))

## [1] -12 -19 9 -13 -14 -17 8 -19 -14

cambiosSigno(x)

## $vector  
## [1] -12 -19 9 -13 -14 -17 8 -19 -14  
##   
## $cambios  
## [1] 4

cambiosSignoPos(x)

## $vector  
## [1] -12 -19 9 -13 -14 -17 8 -19 -14  
##   
## $pos  
## [1] 3 4 7 8

# Sin pasar ningún argumento  
cambiosSigno()

## $vector  
## [1] 5 -1 14 -7 6 -6 -17 -16 6  
##   
## $cambios  
## [1] 6

cambiosSignoPos()

## $vector  
## [1] -14 17 -16 -2 17 -13 1 9 -12  
##   
## $pos  
## [1] 2 3 5 6 7 9

# Ejercicio 3. R4DS.

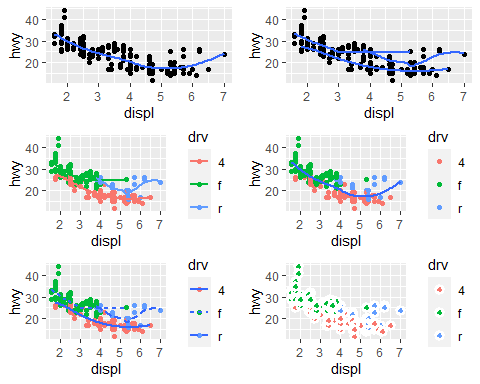
Es recomendable que esta semana del curso hagas al menos una lectura somera de los Capítulos 1 a 5 de [R for Data Science (R4DS), de H. Wickham](https://r4ds.had.co.nz/index.html), con énfasis especial en los Capítulos 3 y 5 (los capítulos 1, 2 y 4 son muy breves). Los siguientes apartados pretenden 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](https://r4ds.had.co.nz/data-visualisation.html#exercises-3).

La base de datos a emplear en este ejercicio es “mpg” que podemos encontrar dentro de la librería tidyverse cargada al inicio.

* Gráficos:

g1 <- ggplot(data = mpg) +   
 geom\_point(mapping = aes(x = displ, y = hwy)) +  
 geom\_smooth(mapping = aes(x = displ, y = hwy), se=FALSE)  
  
g2 <- ggplot(data = mpg) +   
 geom\_point(mapping = aes(x = displ, y = hwy)) +  
 geom\_smooth(mapping = aes(x = displ, y = hwy, group = drv), se= FALSE)  
  
g3 <- ggplot(data = mpg) +   
 geom\_point(mapping = aes(x = displ, y = hwy, color = drv)) +  
 geom\_smooth(mapping = aes(x = displ, y = hwy, group = drv, color = drv), se = FALSE)  
  
g4 <- ggplot(data = mpg) +   
 geom\_point(mapping = aes(x = displ, y = hwy, color = drv)) +  
 geom\_smooth(mapping = aes(x = displ, y = hwy), se = FALSE)  
  
g5 <- ggplot(data = mpg) +   
 geom\_point(mapping = aes(x = displ, y = hwy, color = drv)) +  
 geom\_smooth(mapping = aes(x = displ, y = hwy,group = drv, linetype = drv), se = FALSE)  
  
g6 <- ggplot(data = mpg) +  
 geom\_point(aes(x = displ, y = hwy, fill=drv), shape = 21,color="white",size = 2, stroke = 2)  
  
# Representación de todos los gráficos juntos  
grid.arrange(g1,g2, g3, g4, g5, g6, nrow=3)



* Haz el [ejercicio 1 de la Sección 5.2.4 de R4DS](https://r4ds.had.co.nz/transform.html#exercises-8).

En este ejercicio, la base de datos a utilizar es flights:

head(flights)

## # A tibble: 6 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  
## # ... 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 <dttm>

data("flights") # carga los datos en memoria

Find all flights that

1. Had an arrival delay of two or more hours

Como el retraso está registrado en minutos, se pasan las dos horas a minutos:

filter(flights, 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>

1. Flew to Houston (IAH or HOU)

filter(flights, 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 <dttm>

# Otra forma:  
# filter(flights, dest %in% c("IAH","HOU"))

1. Were operated by United, American, or Delta

filter(flights,carrier=="UA"|carrier=="AA"|carrier=="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>

# Otra forma:  
# filter(flights, carrier %in% c("UA","AA","DL"))

1. Departed in summer (July, August, and September)

filter(flights, 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>

1. Arrived more than two hours late, but didn’t leave late

filter(flights, arr\_delay>120 & dep\_delay <= 0)

## # 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>

1. Were delayed by at least an hour, but made up over 30 minutes in flight

filter(flights, 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>

1. Departed between midnight and 6am (inclusive)

En primer lugar, vemos qué pinta tiene la variable dep\_time para poder analizar de un modo correcto sus valores:

summary(flights$dep\_time)

## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's   
## 1 907 1401 1349 1744 2400 8255

El formato para la media noche es 24:00 y no 0:00, por lo que hay que tenerlo en cuenta a la hora de seleccionar las filas correspondientes al tiempo requerido

filter(flights, 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>