Curso de R

Amparo Garcia

2026-07-01

Table of contents

1 Prefacio

Este curso está dividido en:

- Introducción Básica a R como información adicional
- Tratamiento de datasets

Introducción a R

1.1 Variables

Una variable es como una caja en la cual guardáis una información. Cuando se hace referencia al nombre de la caja estamos utilizando el valor de dentro de la caja,

```
nombre<-'Amparo'
apellido<-'Garcia'
nombre
```

```
[1] "Amparo"
```

1.1.1 Tipo de variables

Numeric -> Tipo de dato numérico

Character -> Tipo de dato carácter

Logical -> Tipo de dato lógico

Integer -> Números enteros sin decimales

```
x <- 3.14
class(x)
```

```
[1] "numeric"
```

```
nombre <- "Amparo"
class(nombre)
```

```
[1] "character"
```

```r
flag <- TRUE
class(flag)
```

```
[1] "logical"
```

1.1.2 Mostrar y Preguntar información

```r
cat("Hola mundo\n")
```

```
Hola mundo
```

```r
nombre <- readline("Ingresa tu nombre: ")
```

```
Ingresa tu nombre:
```

```r
cat("Hola", nombre, "\n")
```

```
Hola
```

1.2 Condicionales

Los condicionales en R permiten que el programa tome decisiones: si se cumple una condición, hace algo; si no, hace otra cosa.

- **if ->** Se usa para ejecutar un bloque de código **solo si** una condición es verdadera.
- **else if->** Se usa para evaluar **otra condición** cuando la primera no se cumple. Puedes tener varios `else if`.
- **else ->** Se ejecuta **cuando ninguna** de las condiciones anteriores es verdadera. No lleva condición.

```r
x <- -5

if (x > 0) {
  cat("x es positivo\n")
} else if (x < 0) {
  cat("x es negativo\n")
} else {
  cat("x es cero\n")
}
```

```
x es negativo
```

1.3 Bucles

Un **bucle** es una instrucción que permite **repetir una acción varias veces sin tener que escribirla una y otra vez**.

La idea es simple: **"Un bucle repite algo hasta que ya no hace falta repetirlo."**

1.3.1 for

Sirve para repetir algo un número conocido de veces.

```r
 for (i in 1:5)
   {
   cat("Ejecución: ", i, " ")
   }
```

```
Ejecución:  1  Ejecución:  2  Ejecución:  3  Ejecución:  4  Ejecución:  5
```

1.3.2 while

Sirve para repetir algo **mientras** se cumpla una condición.

```r
x <- 1
while (x <= 5) {
  cat(x)
  x <- x + 1
  }
```

```
12345
```

1.3.3 Resumen

Los bucles en R sirven para repetir instrucciones sin tener que escribirlas muchas veces. El programa sigue repitiendo hasta que se cumple una condición o hasta que termina la lista de cosas que debe hacer.

1.4 Librerías o paquetes

Una **librería** (o *paquete*) en R es como una **caja de herramientas** que alguien creó para facilitar tareas específicas. Es decir, es un conjunto de funciones extra que puedes instalar para ampliar lo que R se puede hacer.

1.4.1 Cargar el paquete

En otras palabras vendría siendo equivalente a buscar un libro y abrirlo.

```
library(ggplot2)
```

1.4.2 Instalar el paquete

En el caso de no tener instalado el paquete o de que no lo reconozcan cuando lo intentamos utilizar hay que instalar el paquete o la librería: install.packages("nombreDelPaquete")

```
install.packages("ggplot2")
```

```
Warning: package 'ggplot2' is in use and will not be installed
```

1.4.3 Librerías más frecuentes

ggplot2	Gráficos
dplyr	Manipulación de datos
tidyr	Ordenar y limpiar datos
readr	Leer archivos CSV
stringr	Trabajar con texto

2

3 Tratamiento de Datos

3.1 ¿Qué es un dataset?

Un **dataset** es simplemente un **conjunto de datos organizado en forma de tabla**, igual que una hoja de cálculo de Excel.

3.2 Cargar archivos de datos

3.2.1 Archivos Excel

```
library(readxl)
#datos_excel <- read_excel("archivo.xlsx")
```

3.2.2 Archivos csv

```
#datos <- read.csv("archivo.csv")
```

```
coaster_db <- read.csv("C:\\Users\\garci\\Downloads\\coaster_db.csv")
datos <- coaster_db
```

3.3 Información general del dataset

3.3.1 Abrir la tabla como Excel

```
View(datos)
```

3.3.2 Dimensión del dataset (número de filas y de columnas)

```
dim(datos)
```

```
[1] 1087   56
```

El primer valor son las filas y el segundo las columnas

3.3.3 Muestra las primeras filas de un dataset

```
head(datos)
```

```
                          coaster_name         Length            Speed
1                    Switchback Railway 600 ft (180 m) 6 mph (9.7 km/h)
2                     Flip Flap Railway
3 Switchback Railway (Euclid Beach Park)
4           Loop the Loop (Coney Island)
5           Loop the Loop (Young's Pier)
6                         Cannon Coaster
                       Location  Status  Opening.date  Type
1                   Coney Island Removed June 16, 1884  Wood
2                  Sea Lion Park Removed          1895  Wood
3 Cleveland, Ohio, United States  Closed               Other
4                          Other Removed          1901 Steel
5                          Other Removed          1901 Steel
6                   Coney Island Removed          1902  Wood
            Manufacturer Height.restriction       Model       Height Inversions
1 LaMarcus Adna Thompson                    Lift Packed 50 ft (15 m)         NA
2           Lina Beecher                                                      1
3                                                                            NA
4         Edwin Prescott                                                     1
5         Edwin Prescott                                                     1
6   George Francis Meyer                                 40 ft (12 m)         NA
  Lift.launch.system Cost
1            gravity
2
3
4
```

```
6
  Single.rider.line.available Restraint.Style Flash.Pass.available Acceleration
1
2
3
4
5
6
  Restraints Name year_introduced latitude longitude Type_Main
1                            1884  40.5740  -73.9780      Wood
2                            1895  40.5780  -73.9790      Wood
3                            1896  41.5800  -81.5700     Other
4                            1901  40.5745  -73.9780     Steel
5                            1901  39.3538  -74.4342     Steel
6                            1902  40.5750  -73.9800      Wood
  opening_date_clean speed1   speed2 speed1_value speed1_unit speed_mph
1         1884-06-16 6 mph  9.7 km/h            6         mph         6
2         1895-01-01                           NA                    NA
3                                              NA                    NA
4         1901-01-01                           NA                    NA
5         1901-01-01                           NA                    NA
6         1902-01-01                           NA                    NA
  height_value height_unit height_ft Inversions_clean Gforce_clean
1           50          ft        NA                0          2.9
2           NA                    NA                1         12.0
3           NA                    NA                0           NA
4           NA                    NA                1           NA
5           NA                    NA                1           NA
6           40          ft        NA                0           NA
```

3.3.4 Estructura y tipos de datos

- chr -> tipo de dato de texto
- num -> tipo de dato numérico
- int -> tipo de dato entero
- factor -> Un factor es como una columna que solo puede tomar ciertos valores permitidos, llamados niveles.

```
str(datos)
```

```
'data.frame':   1087 obs. of  56 variables:
 $ coaster_name                 : chr  "Switchback Railway" "Flip Flap Railway" "Switchback R
 $ Length                       : chr  "600 ft (180 m)" "" "" "" ...
 $ Speed                        : chr  "6 mph (9.7 km/h)" "" "" "" ...
 $ Location                     : chr  "Coney Island" "Sea Lion Park" "Cleveland, Ohio, Unite
 $ Status                       : chr  "Removed" "Removed" "Closed" "Removed" ...
 $ Opening.date                 : chr  "June 16, 1884" "1895" "" "1901" ...
 $ Type                         : chr  "Wood" "Wood" "Other" "Steel" ...
 $ Manufacturer                 : chr  "LaMarcus Adna Thompson" "Lina Beecher" "" "Edwin Pres
 $ Height.restriction           : chr  "" "" "" "" ...
 $ Model                        : chr  "Lift Packed" "" "" "" ...
 $ Height                       : chr  "50 ft (15 m)" "" "" "" ...
 $ Inversions                   : int  NA 1 NA 1 1 NA NA NA NA NA ...
 $ Lift.launch.system           : chr  "gravity" "" "" "" ...
 $ Cost                         : chr  "" "" "" "" ...
 $ Trains                       : chr  "" "a single car. Riders are arranged 1 across in 2 ro
 $ Park.section                 : chr  "Coney Island Cyclone Site" "" "" "" ...
 $ Duration                     : chr  "1:00" "" "" "" ...
 $ Capacity                     : chr  "1600 riders per hour" "" "" "" ...
 $ G.force                      : chr  "2.9" "12" "" "" ...
 $ Designer                     : chr  "LaMarcus Adna Thompson" "Lina Beecher" "" "Edward A.
 $ Max.vertical.angle           : chr  "30°" "" "" "" ...
 $ Drop                         : chr  "43 ft (13 m)" "" "" "" ...
 $ Soft.opening.date            : chr  "" "" "" "" ...
 $ Fast.Lane.available          : chr  "" "" "" "" ...
 $ Replaced                     : chr  "" "" "" "Switchback Railway" ...
 $ Track.layout                 : chr  "Gravity pulled coaster" "" "" "" ...
 $ Fastrack.available           : chr  "" "" "" "" ...
 $ Soft.opening.date.1          : chr  "" "" "" "" ...
 $ Closing.date                 : chr  "" "1902" "" "1910" ...
 $ Opened                       : chr  "" "" "1895" "" ...
 $ Replaced.by                  : chr  "" "" "" "Giant Racer" ...
 $ Website                      : chr  "" "" "" "" ...
 $ Flash.Pass.Available         : chr  "" "" "" "" ...
 $ Must.transfer.from.wheelchair: chr  "" "" "" "" ...
 $ Theme                        : chr  "" "" "" "" ...
 $ Single.rider.line.available  : chr  "" "" "" "" ...
 $ Restraint.Style              : chr  "" "" "" "" ...
 $ Flash.Pass.available         : chr  "" "" "" "" ...
 $ Acceleration                 : chr  "" "" "" "" ...
 $ Restraints                   : chr  "" "" "" "" ...
 $ Name                         : chr  "" "" "" "" ...
 $ year_introduced              : int  1884 1895 1896 1901 1901 1902 1902 1904 1907 1907 ...
```

```
 $ latitude                    : num  40.6 40.6 41.6 40.6 39.4 ...
 $ longitude                   : num  -74 -74 -81.6 -74 -74.4 ...
 $ Type_Main                   : chr  "Wood" "Wood" "Other" "Steel" ...
 $ opening_date_clean          : chr  "1884-06-16" "1895-01-01" "" "1901-01-
01" ...
 $ speed1                      : chr  "6 mph " "" "" "" ...
 $ speed2                      : chr  "9.7 km/h" "" "" "" ...
 $ speed1_value                : num  6 NA NA NA NA NA 10 NA NA NA ...
 $ speed1_unit                 : chr  "mph" "" "" "" ...
 $ speed_mph                   : num  6 NA NA NA NA NA 10 NA NA NA ...
 $ height_value                : num  50 NA NA NA NA 40 41 NA 60 NA ...
 $ height_unit                 : chr  "ft" "" "" "" ...
 $ height_ft                   : num  NA NA NA NA NA NA NA NA NA NA ...
 $ Inversions_clean            : int  0 1 0 1 1 0 0 0 0 0 ...
 $ Gforce_clean                : num  2.9 12 NA NA NA NA NA NA NA NA ...
```

3.3.5 Nombre de las columnas de un dataset

```
names(datos)
```

```
 [1] "coaster_name"                "Length"
 [3] "Speed"                       "Location"
 [5] "Status"                      "Opening.date"
 [7] "Type"                        "Manufacturer"
 [9] "Height.restriction"          "Model"
[11] "Height"                      "Inversions"
[13] "Lift.launch.system"          "Cost"
[15] "Trains"                      "Park.section"
[17] "Duration"                    "Capacity"
[19] "G.force"                     "Designer"
[21] "Max.vertical.angle"          "Drop"
[23] "Soft.opening.date"           "Fast.Lane.available"
[25] "Replaced"                    "Track.layout"
[27] "Fastrack.available"          "Soft.opening.date.1"
[29] "Closing.date"                "Opened"
[31] "Replaced.by"                 "Website"
[33] "Flash.Pass.Available"        "Must.transfer.from.wheelchair"
[35] "Theme"                       "Single.rider.line.available"
[37] "Restraint.Style"             "Flash.Pass.available"
[39] "Acceleration"                "Restraints"
```

```
[41] "Name"                          "year_introduced"
[43] "latitude"                      "longitude"
[45] "Type_Main"                     "opening_date_clean"
[47] "speed1"                        "speed2"
[49] "speed1_value"                  "speed1_unit"
[51] "speed_mph"                     "height_value"
[53] "height_unit"                   "height_ft"
[55] "Inversions_clean"              "Gforce_clean"
```

3.3.6 Valores únicos de una columna

```
unique(datos$Status)
```

```
 [1] "Removed"
 [2] "Closed"
 [3] "Operating"
 [4] ""
 [5] "Not Currently Operating"
 [6] "In Production"
 [7] "Discontinued"
 [8] "closed for maintenance as of july 30 no reopening date known"
 [9] "Closed in 2021"
[10] "SBNO December 2019"
[11] "Under construction"
[12] "Temporarily Closed"
[13] "SBNO (Standing But Not Operating)"
[14] "Temporarily closed"
[15] "Chapter 7 bankruptcy; rides dismantled and sold; property sold"
[16] "Under Maintenance"
```

3.3.7 Principales Estadísticos

Se pueden obtener las métricas estadísticas de cada columna o de todas las columnas a la vez. En este último caso se haría con la función summary:

```
summary(datos)
```

```
 coaster_name          Length             Speed             Location
 Length:1087        Length:1087        Length:1087        Length:1087
```

```
    latitude        longitude         Type_Main         opening_date_clean
 Min.   :-48.26   Min.   :-123.036   Length:1087        Length:1087
 1st Qu.: 35.03   1st Qu.: -84.552   Class :character   Class :character
 Median : 40.29   Median : -76.654   Mode  :character   Mode  :character
 Mean   : 38.37   Mean   : -41.595
 3rd Qu.: 44.80   3rd Qu.:   2.778
 Max.   : 63.23   Max.   : 153.427
 NA's   :275      NA's   :275
    speed1             speed2           speed1_value    speed1_unit
 Length:1087        Length:1087        Min.   :  5.00   Length:1087
 Class :character   Class :character   1st Qu.: 40.00   Class :character
 Mode  :character   Mode  :character   Median : 50.00   Mode  :character
                                       Mean   : 53.85
                                       3rd Qu.: 63.00
                                       Max.   :240.00
                                       NA's   :150
   speed_mph       height_value     height_unit          height_ft
 Min.   :  5.00   Min.   :   4.00   Length:1087        Min.   : 13.1
 1st Qu.: 37.30   1st Qu.:  44.00   Class :character   1st Qu.: 51.8
 Median : 49.70   Median :  79.00   Mode  :character   Median : 91.2
 Mean   : 48.62   Mean   :  89.58                      Mean   :102.0
 3rd Qu.: 58.00   3rd Qu.: 113.00                      3rd Qu.:131.2
 Max.   :149.10   Max.   :3937.00                      Max.   :377.3
 NA's   :150      NA's   :122                          NA's   :916
 Inversions_clean  Gforce_clean
 Min.   : 0.000   Min.   : 0.800
 1st Qu.: 0.000   1st Qu.: 3.400
 Median : 0.000   Median : 4.000
 Mean   : 1.327   Mean   : 3.824
 3rd Qu.: 2.000   3rd Qu.: 4.500
 Max.   :14.000   Max.   :12.000
                  NA's   :725
```

También se pueden obtener estadísticos de una sola columna ignorando los posibles valores nulos que haya

```
var(datos$latitude, na.rm=TRUE)
```

```
[1] 240.7647
```

```r
mean(datos$latitude, na.rm=TRUE)
```

```
[1] 38.37348
```

3.4 Selección de columnas

Utilizaremos la librería dplyr

```r
library(dplyr)
```

```
Adjuntando el paquete: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```r
coaster_db_sin_height <- coaster_db %>% select(-Height)
head(coaster_db_sin_height)
```

```
                          coaster_name         Length            Speed
1                    Switchback Railway 600 ft (180 m) 6 mph (9.7 km/h)
2                     Flip Flap Railway
3 Switchback Railway (Euclid Beach Park)
4           Loop the Loop (Coney Island)
5           Loop the Loop (Young's Pier)
6                         Cannon Coaster
                       Location  Status  Opening.date  Type
1                   Coney Island Removed June 16, 1884  Wood
2                  Sea Lion Park Removed          1895  Wood
3 Cleveland, Ohio, United States  Closed               Other
4                          Other Removed          1901 Steel
5                          Other Removed          1901 Steel
6                   Coney Island Removed          1902  Wood
```

```
coaster_3_columnas <- coaster_db %>% select(coaster_name, Length, Speed)
head(coaster_3_columnas)
```

```
                          coaster_name         Length            Speed
1                    Switchback Railway 600 ft (180 m) 6 mph (9.7 km/h)
2                     Flip Flap Railway
3 Switchback Railway (Euclid Beach Park)
4           Loop the Loop (Coney Island)
5           Loop the Loop (Young's Pier)
6                         Cannon Coaster
```

3.5 Filtros

Lo más sencillo e intuitivo para hacer filtros es utilizar la librería dplyr

```
coaster_db %>%
  filter(Status %in% c("Closed", "Removed"))%>%filter(Location=='Cedar Point')
```

```
           coaster_name           Length              Speed    Location  Status
1 WildCat (Cedar Point) 1,837 ft (560 m)   40 mph (64 km/h) Cedar Point Removed
2    Disaster Transport 1,932 ft (589 m)   40 mph (64 km/h) Cedar Point Removed
3        Wicked Twister   675 ft (206 m)  72 mph (116 km/h) Cedar Point  Closed
4   Top Thrill Dragster 2,800 ft (850 m) 120 mph (190 km/h) Cedar Point  Closed
  Opening.date                       Type      Manufacturer
1         1979                     Steel Anton Schwarzkopf
2         1985  Steel - Enclosed - Bobsled           Intamin
3  May 5, 2002 Steel - Inverted - Launched           Intamin
4  May 4, 2003            Steel - Launched           Intamin
       Height.restriction                   Model         Height Inversions
1           48 in (122 cm)            Wildcat/65m   50 ft (15 m)          0
2           46 in (117 cm)              Swiss Bob   63 ft (19 m)          0
3    52-78 in (132-198 cm) Twisted Impulse Coaster  215 ft (66 m)          0
4 52-78[1] in (132-198 cm)     Accelerator Coaster 420 ft (130 m)          0
  Lift.launch.system
1        Chain lift
2             Chain
3   LIM Launch track
4   Hydraulic Launch
                                                                  Cost
1
```

```
  Restraint.Style Flash.Pass.available
1
2
3
4
                             Acceleration Restraints Name year_introduced
1                                                                    1979
2                                                                    1985
3                                                                    2002
4 0 to 120 mph (0 to 193 km/h) in 4 seconds                          2003
  latitude longitude Type_Main opening_date_clean   speed1   speed2
1  41.4817  -82.6850     Steel         1979-01-01  40 mph   64 km/h
2  41.4811  -82.6794     Steel         1985-01-01  40 mph   64 km/h
3  41.4820  -82.6799     Steel         2002-05-05  72 mph  116 km/h
4  41.4840  -82.6862     Steel         2003-05-04 120 mph  190 km/h
  speed1_value speed1_unit speed_mph height_value height_unit height_ft
1           40         mph        40           50          ft        NA
2           40         mph        40           63          ft        NA
3           72         mph        72          215          ft        NA
4          120         mph       120          420          ft        NA
  Inversions_clean Gforce_clean
1                0           NA
2                0          2.7
3                0           NA
4                0           NA
```

3.5.1 Estadísticas por grupos

```
mean(datos$latitude, na.rm=TRUE)
```

```
[1] 38.37348
```

```
datos %>%
  group_by(Status) %>%
  summarise(media = mean(latitude, na.rm = TRUE), maximo=max(latitude, na.rm = TRUE))
```

```
Warning: There were 2 warnings in `summarise()`.
The first warning was:
i In argument: `maximo = max(latitude, na.rm = TRUE)`.
i In group 5: `Status = "Discontinued"`.
```

```
Caused by warning in `max()`:
! ningun argumento finito para max; retornando -Inf
i Run `dplyr::last_dplyr_warnings()` to see the 1 remaining warning.
```

```
# A tibble: 16 x 3
   Status                                                          media maximo
   <chr>                                                           <dbl>  <dbl>
 1 ""                                                               34.1   53.0
 2 "Chapter 7 bankruptcy; rides dismantled and sold; property sold"  33.7   33.7
 3 "Closed"                                                         39.4   54.2
 4 "Closed in 2021"                                                 54.2   54.2
 5 "Discontinued"                                                  NaN   -
Inf
 6 "In Production"                                                 NaN   -
Inf
 7 "Not Currently Operating"                                        49.3   49.3
 8 "Operating"                                                      39.6   63.2
 9 "Removed"                                                        35.2   60.2
10 "SBNO (Standing But Not Operating)"                              38.1   38.1
11 "SBNO December 2019"                                            -27.9  -
27.9
12 "Temporarily Closed"                                             43.0   43.0
13 "Temporarily closed"                                             42.4   42.4
14 "Under Maintenance"                                              22.3   22.3
15 "Under construction"                                             16.4   37.2
16 "closed for maintenance as of july 30 no reopening date known"   53.5   53.5
```

3.6 Crear una columna en función del valor de otra

```
datos$es_madera <- ifelse(datos$Type =="Wood", 1, 0)
View(datos)
```

```
prop.table(table(datos$es_madera))
```

```
       0        1
0.849126 0.150874
```

3.7 Valores nulos

3.7.1 Conocer si hay valores nulos

```
is.na(coaster_db)
```

```
      coaster_name Length Speed Location Status Opening.date  Type
 [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
 [7,]        FALSE  FALSE FALSE    FALSE  FALSE        FALSE FALSE
 [8,]        FALSE  FALSE FALSE    FALSE  FALSE        FALSE FALSE
 [9,]        FALSE  FALSE FALSE    FALSE  FALSE        FALSE FALSE
[10,]        FALSE  FALSE FALSE    FALSE  FALSE        FALSE FALSE
[11,]        FALSE  FALSE FALSE    FALSE  FALSE        FALSE FALSE
[12,]        FALSE  FALSE FALSE    FALSE  FALSE        FALSE FALSE
[13,]        FALSE  FALSE FALSE    FALSE  FALSE        FALSE FALSE
[14,]        FALSE  FALSE FALSE    FALSE  FALSE        FALSE FALSE
[15,]        FALSE  FALSE FALSE    FALSE  FALSE        FALSE FALSE
[16,]        FALSE  FALSE FALSE    FALSE  FALSE        FALSE FALSE
[17,]        FALSE  FALSE FALSE    FALSE  FALSE        FALSE FALSE
[18,]        FALSE  FALSE FALSE    FALSE  FALSE        FALSE FALSE
[19,]        FALSE  FALSE FALSE    FALSE  FALSE        FALSE FALSE
[20,]        FALSE  FALSE FALSE    FALSE  FALSE        FALSE FALSE
[21,]        FALSE  FALSE FALSE    FALSE  FALSE        FALSE FALSE
[22,]        FALSE  FALSE FALSE    FALSE  FALSE        FALSE FALSE
[23,]        FALSE  FALSE FALSE    FALSE  FALSE        FALSE FALSE
[24,]        FALSE  FALSE FALSE    FALSE  FALSE        FALSE FALSE
[25,]        FALSE  FALSE FALSE    FALSE  FALSE        FALSE FALSE
[26,]        FALSE  FALSE FALSE    FALSE  FALSE        FALSE FALSE
[27,]        FALSE  FALSE FALSE    FALSE  FALSE        FALSE FALSE
[28,]        FALSE  FALSE FALSE    FALSE  FALSE        FALSE FALSE
[29,]        FALSE  FALSE FALSE    FALSE  FALSE        FALSE FALSE
[30,]        FALSE  FALSE FALSE    FALSE  FALSE        FALSE FALSE
[31,]        FALSE  FALSE FALSE    FALSE  FALSE        FALSE FALSE
[32,]        FALSE  FALSE FALSE    FALSE  FALSE        FALSE FALSE
[33,]        FALSE  FALSE FALSE    FALSE  FALSE        FALSE FALSE
[34,]        FALSE  FALSE FALSE    FALSE  FALSE        FALSE FALSE
```

```
[1066,]     FALSE            FALSE         TRUE
[1067,]      TRUE            FALSE        FALSE
[1068,]      TRUE            FALSE        FALSE
[1069,]      TRUE            FALSE         TRUE
[1070,]      TRUE            FALSE         TRUE
[1071,]      TRUE            FALSE         TRUE
[1072,]      TRUE            FALSE         TRUE
[1073,]      TRUE            FALSE         TRUE
[1074,]      TRUE            FALSE        FALSE
[1075,]      TRUE            FALSE         TRUE
[1076,]      TRUE            FALSE         TRUE
[1077,]      TRUE            FALSE         TRUE
[1078,]      TRUE            FALSE        FALSE
[1079,]      TRUE            FALSE         TRUE
[1080,]      TRUE            FALSE         TRUE
[1081,]      TRUE            FALSE         TRUE
[1082,]     FALSE            FALSE         TRUE
[1083,]      TRUE            FALSE        FALSE
[1084,]      TRUE            FALSE         TRUE
[1085,]      TRUE            FALSE        FALSE
[1086,]      TRUE            FALSE         TRUE
[1087,]      TRUE            FALSE         TRUE
```

```
any(is.na(coaster_db)) #¿Hay valores nulos?
```

```
[1] TRUE
```

```
sum(is.na(coaster_db)) #Cuantos nulos hay
```

```
[1] 2768
```

```
colSums(is.na(coaster_db)) #Cuantos nulos hay por columna
```

```
coaster_name                       Length
           0                            0
       Speed                     Location
           0                            0
      Status                Opening.date
           0                            0
        Type                 Manufacturer
```

```
      speed_mph                  height_value
            150                           122
   height_unit                     height_ft
              0                           916
Inversions_clean                 Gforce_clean
              0                           725
```

3.7.2 Tratamiento de valores nulos

Si en el dataset hay valores nulos existen diversas formas de actuar:

- **Eliminar la columna con los nulos.** Si la columna no es importante o todos los datos son nulos, lo mejor es eliminar la columna.
- **Eliminar todas las filas con valores nulos u omitirlos.** Si el número de filas con nulos no son demasiadas entonces podemos optar por eliminar estas filas.
- **Reemplazar los valores nulos por otros valores.** Cuando no se dan ninguna de las dos situaciones anteriores generalmente se reemplazar por un valor que puede ser por ejemplo 0, o la media o la mediana.

3.7.2.1 Eliminar una columna

```
library(dplyr)
coaster_db_sin_height <- coaster_db %>% select(-Height)
```

3.7.2.2 Omitir los valores nulos

```
#Eliminar todas las filas que tengan al menos un NA
coaster_db_sin_na <- na.omit(coaster_db)
```

3.7.2.3 Reemplazar los valores nulos por otro valor

```
coaster_db[is.na(coaster_db)] <- 0 #Todos los valores nulos se reemplazan por 0
coaster_db$Inversions[is.na(coaster_db$Inversions )] <- '0' #Todos los valores nulos de la co
```

3.7.3 Convertir una columna de un tipo de dato a otro

3.7.3.1 De str a factor

Un factor en R es un tipo de dato especial que se usa para representar variables categóricas, es decir, variables que toman un **conjunto limitado de valores posibles** (categorías o niveles). Cada uno de estos valores posibles son los "levels".

En el ejemplo de abajo la expresión

```
Factor w/ 990 levels
```

indica que la columna ya es de tipo factor y tiene 990 posibles valores.

```
coaster_db$coaster_name <- as.factor(coaster_db$coaster_name)
str(coaster_db)
```

```
'data.frame':   1087 obs. of  56 variables:
 $ coaster_name                : Factor w/ 990 levels "10 Inversion Roller Coaster",..: 737
 $ Length                      : chr  "600 ft (180 m)" "" "" "" ...
 $ Speed                       : chr  "6 mph (9.7 km/h)" "" "" "" ...
 $ Location                    : chr  "Coney Island" "Sea Lion Park" "Cleveland, Ohio, Unite
 $ Status                      : chr  "Removed" "Removed" "Closed" "Removed" ...
 $ Opening.date                : chr  "June 16, 1884" "1895" "" "1901" ...
 $ Type                        : chr  "Wood" "Wood" "Other" "Steel" ...
 $ Manufacturer                : chr  "LaMarcus Adna Thompson" "Lina Beecher" "" "Edwin Pres
 $ Height.restriction          : chr  "" "" "" "" ...
 $ Model                       : chr  "Lift Packed" "" "" "" ...
 $ Height                      : chr  "50 ft (15 m)" "" "" "" ...
 $ Inversions                  : chr  "0" "1" "0" "1" ...
 $ Lift.launch.system          : chr  "gravity" "" "" "" ...
 $ Cost                        : chr  "" "" "" "" ...
 $ Trains                      : chr  "" "a single car. Riders are arranged 1 across in 2 ro
 $ Park.section                : chr  "Coney Island Cyclone Site" "" "" "" ...
 $ Duration                    : chr  "1:00" "" "" "" ...
 $ Capacity                    : chr  "1600 riders per hour" "" "" "" ...
 $ G.force                     : chr  "2.9" "12" "" "" ...
 $ Designer                    : chr  "LaMarcus Adna Thompson" "Lina Beecher" "" "Edward A.
 $ Max.vertical.angle          : chr  "30°" "" "" "" ...
 $ Drop                        : chr  "43 ft (13 m)" "" "" "" ...
 $ Soft.opening.date           : chr  "" "" "" "" ...
 $ Fast.Lane.available         : chr  "" "" "" "" ...
```

```
 $ Replaced                     : chr  "" "" "" "Switchback Railway" ...
 $ Track.layout                 : chr  "Gravity pulled coaster" "" "" "" ...
 $ Fastrack.available           : chr  "" "" "" "" ...
 $ Soft.opening.date.1          : chr  "" "" "" "" ...
 $ Closing.date                 : chr  "" "1902" "" "1910" ...
 $ Opened                       : chr  "" "" "1895" "" ...
 $ Replaced.by                  : chr  "" "" "" "Giant Racer" ...
 $ Website                      : chr  "" "" "" "" ...
 $ Flash.Pass.Available         : chr  "" "" "" "" ...
 $ Must.transfer.from.wheelchair: chr  "" "" "" "" ...
 $ Theme                        : chr  "" "" "" "" ...
 $ Single.rider.line.available  : chr  "" "" "" "" ...
 $ Restraint.Style              : chr  "" "" "" "" ...
 $ Flash.Pass.available         : chr  "" "" "" "" ...
 $ Acceleration                 : chr  "" "" "" "" ...
 $ Restraints                   : chr  "" "" "" "" ...
 $ Name                         : chr  "" "" "" "" ...
 $ year_introduced              : int  1884 1895 1896 1901 1901 1902 1902 1904 1907 1907 ...
 $ latitude                     : num  40.6 40.6 41.6 40.6 39.4 ...
 $ longitude                    : num  -74 -74 -81.6 -74 -74.4 ...
 $ Type_Main                    : chr  "Wood" "Wood" "Other" "Steel" ...
 $ opening_date_clean           : chr  "1884-06-16" "1895-01-01" "" "1901-01-
01" ...
 $ speed1                       : chr  "6 mph " "" "" "" ...
 $ speed2                       : chr  "9.7 km/h" "" "" "" ...
 $ speed1_value                 : num  6 0 0 0 0 0 10 0 0 0 ...
 $ speed1_unit                  : chr  "mph" "" "" "" ...
 $ speed_mph                    : num  6 0 0 0 0 0 10 0 0 0 ...
 $ height_value                 : num  50 0 0 0 0 40 41 0 60 0 ...
 $ height_unit                  : chr  "ft" "" "" "" ...
 $ height_ft                    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Inversions_clean             : int  0 1 0 1 1 0 0 0 0 0 ...
 $ Gforce_clean                 : num  2.9 12 0 0 0 0 0 0 0 0 ...
```

Podemos incluso cambiar todas las variables de tipo str a factores sin ir una a una con dplyr:

```
library(dplyr)
coaster_db_factor <- coaster_db %>% mutate(across(where(is.character), as.factor))
str(coaster_db_factor)
```

```
'data.frame':   1087 obs. of  56 variables:
 $ coaster_name                 : Factor w/ 990 levels "10 Inversion Roller Coaster",..: 737
```

```
$ longitude                    : num  -74 -74 -81.6 -74 -74.4 ...
$ Type_Main                    : Factor w/ 3 levels "Other","Steel",..: 3 3 1 2 2 3 3 1 1 1
$ opening_date_clean           : Factor w/ 603 levels "","1884-06-16",..: 2 3 1 4 4 5 5 1 6
$ speed1                       : Factor w/ 226 levels "","10 mph ","100 km/h ",..: 150 1 1 1
$ speed2                       : Factor w/ 230 levels "","100 km/h",..: 212 1 1 1 1 1 49 1 1
$ speed1_value                 : num  6 0 0 0 0 0 10 0 0 0 ...
$ speed1_unit                  : Factor w/ 3 levels "","km/h","mph": 3 1 1 1 1 1 3 1 1 1 ...
$ speed_mph                    : num  6 0 0 0 0 0 10 0 0 0 ...
$ height_value                 : num  50 0 0 0 0 40 41 0 60 0 ...
$ height_unit                  : Factor w/ 3 levels "","ft","m": 2 1 1 1 1 2 2 1 2 1 ...
$ height_ft                    : num  0 0 0 0 0 0 0 0 0 0 ...
$ Inversions_clean             : int  0 1 0 1 1 0 0 0 0 0 ...
$ Gforce_clean                 : num  2.9 12 0 0 0 0 0 0 0 0 ...
```

Al haberlas transformado en factores ahora podemos hacer operaciones que antes como texto no podía: regresiones.

Sin embargo, para otras cosas como correlaciones se necesita pasar estos factores a numéricos. El problema es que si pasamos directamente el factor a numérico entonces da códigos ficticios, por lo que lo ideal es hacer convertir de texto a factor (para que haya esos números para cada valor de la columna) , de factor a textoy de texto a numérico.

3.7.3.2 Conversión de texto a numérico para regresión logística

```
#install.packages("fastDummies")
library(fastDummies)
coaster_encoded <- dummy_cols(coaster_db, remove_first_dummy = TRUE)
View(coaster_encoded)
```

Si miramos bien veremos que después de las columnas que yo estaban se han generado muchísimas columnas con 1 o 0 en las celdas y que los nombres de las columnas son las columnas originales.valor_celda.

3.7.3.3 Conversión de texto a número para árboles

```
coaster_db_factor$Location <- as.numeric(as.factor(coaster_db_factor$Location))
head(coaster_db_factor)
```

```
2         1895-01-01                          0                    0
3                                             0                    0
4         1901-01-01                          0                    0
5         1901-01-01                          0                    0
6         1902-01-01                          0                    0
  height_value height_unit height_ft Inversions_clean Gforce_clean
1           50          ft         0                0          2.9
2            0                     0                1         12.0
3            0                     0                0          0.0
4            0                     0                1          0.0
5            0                     0                1          0.0
6           40          ft         0                0          0.0
```

Si vemos la columna Location son números en vez de los textos comentados anteriormente

4 Modelos de Aprendizaje Automático

4.1 Carga de Datos

```
datos <- read.csv("C:\\Users\\garci\\Downloads\\weatherAUS.csv")
View(datos)
```

```
dim(datos)
```

```
[1] 145460     23
```

```
colSums(is.na(datos)) #Cuantos nulos hay por columna
```

```
         Date      Location       MinTemp       MaxTemp      Rainfall 
            0             0          1485          1261          3261 
  Evaporation      Sunshine  WindGustDir WindGustSpeed    WindDir9am 
        62790         69835         10326         10263         10566 
   WindDir3pm  WindSpeed9am  WindSpeed3pm   Humidity9am   Humidity3pm 
         4228          1767          3062          2654          4507 
  Pressure9am   Pressure3pm      Cloud9am      Cloud3pm       Temp9am 
        15065         15028         55888         59358          1767 
     Temp3pm     RainToday  RainTomorrow 
         3609          3261          3267 
```

```
datos[is.na(datos)] <- 0 #Todos los valores nulos se reemplazan por 0
```

4.2 Covarianza

```
#Seleccionamos las columnas numéricas y luego calculamos la covarianza. Importante utilizar e
library(dplyr)
```

```
Adjuntando el paquete: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```r
datos%>% select(where(is.numeric)) %>% cov(use = "complete.obs")
```

```
                 MinTemp    MaxTemp   Rainfall Evaporation    Sunshine
MinTemp        42.025305  34.700388  5.6064706    9.699622   4.0734364
MaxTemp        34.700388  54.875627 -4.1684137   11.924144   8.3834016
Rainfall        5.606471  -4.168414 70.3882566   -1.871999  -3.7892085
Evaporation     9.699622  11.924144 -1.8719988   17.331151   9.8175227
Sunshine        4.073436   8.383402 -3.7892085    9.817523  21.9097509
WindGustSpeed  14.433491  11.991770 12.9042330    9.249295   4.3990242
WindSpeed9am   10.509037   1.624993  6.3586564    6.341469   4.3034131
WindSpeed3pm   10.296978   4.555563  3.9751369    4.864572   5.3446952
Humidity9am   -18.709724 -54.260376 35.2616470  -27.109254 -23.6046482
Humidity3pm     3.134374 -65.925833 42.1583097  -23.128581 -25.5621778
Pressure9am   311.415557 334.912863 -1.6357563  285.981724 383.1855283
Pressure3pm   305.942284 329.302598 -1.5904896  284.229002 380.1317186
Cloud9am        3.322552  -2.162932  3.9570911    2.029696   0.9143533
Cloud3pm        2.491523  -1.782500  3.2399710    2.182235   1.1559848
Temp9am        38.838148  43.028243  0.7075461   10.916124   6.5382310
Temp3pm        31.375288  49.819151 -4.6240761   11.242139   9.7945213
               WindGustSpeed WindSpeed9am WindSpeed3pm Humidity9am Humidity3pm
MinTemp           14.4334906    10.509037    10.296978   -18.70972    3.134374
MaxTemp           11.9917700     1.624993     4.555563   -54.26038  -65.925833
Rainfall          12.9042330     6.358656     3.975137    35.26165   42.158310
Evaporation        9.2492953     6.341469     4.864572   -27.10925  -23.128581
Sunshine           4.3990242     4.303413     5.344695   -23.60465  -25.562178
WindGustSpeed    277.1980289    78.904126    92.209117   -51.84700    2.008985
WindSpeed9am      78.9041262    80.885017    42.323986   -39.61784   -2.167010
WindSpeed3pm      92.2091169    42.323986    83.156262   -19.89467   17.548225
Humidity9am     -51.8469991   -39.617845   -19.894667   440.49063  291.874105
Humidity3pm       2.0089846    -2.167010    17.548225   291.87411  498.823442
```

```
Pressure9am     945.8602464   373.107116   562.047604   303.55605  342.538518
Pressure3pm     949.8216050   365.418804   584.590443   242.70753  413.660630
Cloud9am          0.6698416     2.864326     2.487489    16.20441   18.976770
Cloud3pm          3.2275700     3.306124     2.886863    10.87797   21.497233
Temp9am          14.1960757     8.343949    10.109498   -42.23540  -25.255206
Temp3pm          15.1171235     1.776496     7.715198   -51.34786  -50.959805
               Pressure9am Pressure3pm    Cloud9am    Cloud3pm     Temp9am
MinTemp         311.415557   305.94228   3.3225518   2.4915225  38.8381481
MaxTemp         334.912863   329.30260  -2.1629318  -1.7825002  43.0282428
Rainfall         -1.635756    -1.59049   3.9570911   3.2399710   0.7075461
Evaporation     285.981724   284.22900   2.0296957   2.1822355  10.9161240
Sunshine        383.185528   380.13172   0.9143533   1.1559848   6.5382310
WindGustSpeed   945.860246   949.82161   0.6698416   3.2275700  14.1960757
WindSpeed9am    373.107116   365.41880   2.8643264   3.3061240   8.3439490
WindSpeed3pm    562.047604   584.59044   2.4874886   2.8868628  10.1094975
Humidity9am     303.556052   242.70753  16.2044069  10.8779735 -42.2353983
Humidity3pm     342.538518   413.66063  18.9767705  21.4972328 -25.2552062
Pressure9am   96193.817235 94138.95995 165.9358588 178.3719549 346.4770429
Pressure3pm   94138.959950 95533.15534 165.9189521 180.7916936 326.1313532
Cloud9am        165.935859   165.91895   9.8128036   6.6875151   0.5297990
Cloud3pm        178.371955   180.79169   6.6875151   9.2934711   0.6678503
Temp9am         346.477043   326.13135   0.5297990   0.6678503  45.0566817
Temp3pm         415.793021   434.62202  -2.4569643  -1.1144636  39.5009613
                 Temp3pm
MinTemp         31.375288
MaxTemp         49.819151
Rainfall        -4.624076
Evaporation     11.242139
Sunshine         9.794521
WindGustSpeed   15.117123
WindSpeed9am     1.776496
WindSpeed3pm     7.715198
Humidity9am    -51.347859
Humidity3pm    -50.959805
Pressure9am    415.793021
Pressure3pm    434.622022
Cloud9am        -2.456964
Cloud3pm        -1.114464
Temp9am         39.500961
Temp3pm         58.299277
```

- Covarianza alta entre **MinTemp y MaxTemp** (33.8): como era de esperar, las temperaturas mínima y máxima están fuertemente relacionadas. Ambas tienen covarianza

positiva con **Evaporation** y **Sunshine**, lo que sugiere que días más cálidos tienden a ser más soleados y con mayor evaporación.

- Covarianza negativa entre **Lluvia (Rainfall)** con **Sunshine** (–6.4): más lluvia implica menos sol y covarianza positiva con **Humidity3pm** (39): días lluviosos suelen tener más humedad por la tarde.
- **WindGustSpeed**, **WindSpeed9am** y **WindSpeed3pm** tienen covarianzas muy altas entre sí (70–79): los distintos tipos de velocidad del viento están fuertemente correlacionados. También tienen covarianza positiva con **Evaporation**, lo que sugiere que el viento favorece la evaporación.
- **Humidity9am** y **Humidity3pm** tienen covarianza positiva con **Rainfall** y negativa con **Sunshine** y **Evaporation**: más humedad implica menos sol y menos evaporación.
- **Pressure9am** y **Pressure3pm** tienen covarianzas negativas con casi todo: presión baja suele coincidir con condiciones más extremas (más lluvia, más humedad).
- **Cloud9am** tiene covarianza negativa con **Sunshine** (–7.17): más nubes, menos sol.

También se puede hacer la covarianza de dos columnas:

```
cov(datos$MinTemp, datos$MaxTemp, use = "complete.obs")
```

```
[1] 34.70039
```

4.3

4.4 Correlación

La correlación es un valor entre -1 y 1. De forma que:

- Correlación entre -1 y -0.5 -> correlación negativa fuerte
- Correlación entre -0.5 y 0 -> correlación negativa débil
- Correlación =0 -> independiente
- Correlación entre 0 y 0.5 -> correlación positiva débil
- Correlación entre 0.5 y 1 -> correlación positiva fuerte

```
datos%>% select(where(is.numeric)) %>%cor(use = "complete.obs")
```

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine
MinTemp	1.00000000	0.72258534	0.1030822777	0.35940627	0.13424150
MaxTemp	0.72258534	1.00000000	-0.0670704098	0.38665512	0.24177523
Rainfall	0.10308228	-0.06707041	1.0000000000	-0.05359716	-0.09648942
Evaporation	0.35940627	0.38665512	-0.0535971568	1.00000000	0.50381314
Sunshine	0.13424150	0.24177523	-0.0964894155	0.50381314	1.00000000
WindGustSpeed	0.13372754	0.09722959	0.0923819283	0.13344425	0.05644722
WindSpeed9am	0.18024923	0.02439089	0.0842715946	0.16937220	0.10222574
WindSpeed3pm	0.17418345	0.06743802	0.0519582490	0.12813963	0.12521511
Humidity9am	-0.13751295	-0.34899943	0.2002553620	-0.31026696	-0.24027605
Humidity3pm	0.02164821	-0.39846709	0.2249881570	-0.24874929	-0.24451511
Pressure9am	0.15488569	0.14577012	-0.0006286297	0.22148841	0.26394704
Pressure3pm	0.15268874	0.14382300	-0.0006133434	0.22089080	0.26274734
Cloud9am	0.16361374	-0.09320871	0.1505668577	0.15563980	0.06235895
Cloud3pm	0.12607251	-0.07893159	0.1266782459	0.17194871	0.08101108
Temp9am	0.89253134	0.86533499	0.0125639168	0.39063860	0.20809514
Temp3pm	0.63387072	0.88079530	-0.0721843880	0.35367443	0.27405196

	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm
MinTemp	0.133727544	0.18024923	0.17418345	-0.13751295	0.021648211
MaxTemp	0.097229595	0.02439089	0.06743802	-0.34899943	-0.398467088
Rainfall	0.092381928	0.08427159	0.05195825	0.20025536	0.224988157
Evaporation	0.133444251	0.16937220	0.12813963	-0.31026696	-0.248749292
Sunshine	0.056447221	0.10222574	0.12521511	-0.24027605	-0.244515111
WindGustSpeed	1.000000000	0.52695145	0.60733920	-0.14837482	0.005402666
WindSpeed9am	0.526951453	1.00000000	0.51606576	-0.20988842	-0.010788311
WindSpeed3pm	0.607339202	0.51606576	1.00000000	-0.10394913	0.086161307
Humidity9am	-0.148374825	-0.20988842	-0.10394913	1.00000000	0.622664066
Humidity3pm	0.005402666	-0.01078831	0.08616131	0.62266407	1.000000000
Pressure9am	0.183171678	0.13375989	0.19872479	0.04663339	0.049449557
Pressure3pm	0.184573741	0.13145581	0.20740880	0.03741432	0.059923021
Cloud9am	0.012843414	0.10166979	0.08707973	0.24647238	0.271239118
Cloud3pm	0.063590410	0.12058579	0.10384598	0.17001635	0.315733131
Temp9am	0.127026296	0.13821590	0.16515909	-0.29979795	-0.168460424
Temp3pm	0.118916606	0.02587014	0.11080721	-0.32042207	-0.298829002

	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am
MinTemp	0.1548856872	0.1526887384	0.16361374	0.12607251	0.89253134
MaxTemp	0.1457701172	0.1438230020	-0.09320871	-0.07893159	0.86533499
Rainfall	-0.0006286297	-0.0006133434	0.15056686	0.12667825	0.01256392
Evaporation	0.2214884050	0.2208907983	0.15563980	0.17194871	0.39063860
Sunshine	0.2639470418	0.2627473398	0.06235895	0.08101108	0.20809514
WindGustSpeed	0.1831716782	0.1845737407	0.01284341	0.06359041	0.12702630
WindSpeed9am	0.1337598932	0.1314558118	0.10166979	0.12058579	0.13821590
WindSpeed3pm	0.1987247877	0.2074087960	0.08707973	0.10384598	0.16515909

```
Humidity9am    0.0466333892  0.0374143195  0.24647238  0.17001635 -0.29979795
Humidity3pm    0.0494495573  0.0599230211  0.27123912  0.31573313 -0.16846042
Pressure9am    1.0000000000  0.9820164329  0.17079302  0.18865313  0.16642604
Pressure3pm    0.9820164329  1.0000000000  0.17136510  0.19187237  0.15719397
Cloud9am       0.1707930185  0.1713651010  1.00000000  0.70029213  0.02519620
Cloud3pm       0.1886531310  0.1918723686  0.70029213  1.00000000  0.03263702
Temp9am        0.1664260411  0.1571939725  0.02519620  0.03263702  1.00000000
Temp3pm        0.1755787813  0.1841632968 -0.10272371 -0.04787903  0.77071973
                  Temp3pm
MinTemp        0.63387072
MaxTemp        0.88079530
Rainfall      -0.07218439
Evaporation    0.35367443
Sunshine       0.27405196
WindGustSpeed  0.11891661
WindSpeed9am   0.02587014
WindSpeed3pm   0.11080721
Humidity9am   -0.32042207
Humidity3pm   -0.29882900
Pressure9am    0.17557878
Pressure3pm    0.18416330
Cloud9am      -0.10272371
Cloud3pm      -0.04787903
Temp9am        0.77071973
Temp3pm        1.00000000
```

- **MinTemp y MaxTemp**: correlación alta (**0.75**) → días cálidos tienen mínimas y máximas elevadas. **MaxTemp y Evaporation**: fuerte correlación (**0.65**) → más calor, más evaporación. **MaxTemp y Sunshine**: moderada (**0.47**) → días más cálidos tienden a ser más soleados.
- **Rainfall y Sunshine**: correlación negativa (–**0.24**) → más lluvia, menos sol. **Rainfall y Humidity3pm**: positiva (**0.28**) → días lluviosos suelen tener más humedad por la tarde.
- **WindGustSpeed y WindSpeed9am**: alta correlación (**0.61**). **WindGustSpeed y WindSpeed3pm**: aún más alta (**0.69**) → Las distintas medidas de viento están fuertemente relacionadas.
- **Humidity9am y MaxTemp**: correlación negativa (–**0.50**). **Humidity3pm y Sunshine**: muy negativa (–**0.63**) → Más humedad suele coincidir con menos sol y menos temperatura.
- **Pressure9am y MinTemp**: correlación negativa (–**0.48**). **Pressure3pm y MaxTemp**: también negativa (–**0.46**) → Presión baja suele coincidir con temperaturas más extremas.

- **Cloud9am y Sunshine**: correlación muy negativa (**–0.68**) → Más nubes por la mañana, menos sol durante el día.

4.5 Regresión Lineal

Sirve para predecir **valores numéricos** en base a otras columnas. Por ejemplo queremos predecir la cantidad de lluvia en base a

4.5.1 Seleccionar las columnas para la regresión lineal

```
names(datos%>%select(where(is.numeric)))
```

```
 [1] "MinTemp"       "MaxTemp"       "Rainfall"      "Evaporation"
 [5] "Sunshine"      "WindGustSpeed" "WindSpeed9am"  "WindSpeed3pm"
 [9] "Humidity9am"   "Humidity3pm"   "Pressure9am"   "Pressure3pm"
[13] "Cloud9am"      "Cloud3pm"      "Temp9am"       "Temp3pm"
```

```
datos_regresion <- datos%>%select("MinTemp","MaxTemp","Rainfall","Evaporation","Sunshine","Wi
```

4.5.2 Dividir el dataset en entrenamiento y testeo

```
set.seed(123) # para reproducibilidad y aleatoriedad
# Proporción de entrenamiento (80%). Seleccionamos un 80% de los datos para el entrenamiento
indice <- sample(1:nrow(datos_regresion), size = 0.8 * nrow(datos_regresion))
train <- datos_regresion[indice, ] # datos de entrenamiento
test <- datos_regresion[-indice, ] # datos de testeo
```

```
dim(test)
```

```
[1] 29092    16
```

```
dim(train)
```

```
[1] 116368     16
```

```r
dim(datos_regresion)
```

```
[1] 145460     16
```

```r
#El número de filas del conjunto de entrenamiento y de testeo al sumarlos deben dar lo mismo
```

4.5.3 Entrenar el modelo de regresión lineal

Aplicamos el modelo de regresión lineal al conjunto de entrenamiento. La columna que se quiere predecir es Rainfall y el resto de columnas se añaden después de ~ se suman entre ellas.

```r
modelo <- lm(Rainfall ~ MinTemp + MaxTemp + Evaporation + Sunshine + WindGustSpeed + WindSpee
```

4.5.4 Analizar el modelo

Podemos obtener el resumen del modelo de la siguiente forma:

```r
summary(modelo)
```

```
Call:
lm(formula = Rainfall ~ MinTemp + MaxTemp + Evaporation + Sunshine +
    WindGustSpeed + WindSpeed9am + WindSpeed3pm + Humidity9am +
    Humidity3pm + Pressure9am + Pressure3pm + Cloud9am + Cloud3pm +
    Temp9am + Temp3pm, data = train)

Residuals:
   Min     1Q Median     3Q    Max
-10.51  -2.98  -1.28   0.61 366.41

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)   -3.8588152  0.1711068 -22.552  < 2e-16 ***
MinTemp        0.2723961  0.0092329  29.503  < 2e-16 ***
MaxTemp       -0.0700561  0.0100109  -6.998 2.61e-12 ***
Evaporation   -0.0475538  0.0073113  -6.504 7.84e-11 ***
Sunshine      -0.0578579  0.0061120  -9.466  < 2e-16 ***
WindGustSpeed  0.0546061  0.0019087  28.609  < 2e-16 ***
WindSpeed9am   0.0539864  0.0033826  15.960  < 2e-16 ***
```

```
WindSpeed3pm  -0.0420854  0.0034884 -12.064  < 2e-16 ***
Humidity9am    0.0565167  0.0016307  34.658  < 2e-16 ***
Humidity3pm    0.0224355  0.0017139  13.090  < 2e-16 ***
Pressure9am   -0.0015334  0.0004127  -3.715 0.000203 ***
Pressure3pm    0.0005058  0.0004142   1.221 0.221999
Cloud9am       0.1367069  0.0111551  12.255  < 2e-16 ***
Cloud3pm       0.0568619  0.0112947   5.034 4.80e-07 ***
Temp9am       -0.0324232  0.0116485  -2.783 0.005379 **
Temp3pm       -0.0531099  0.0067913  -7.820 5.32e-15 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.005 on 116352 degrees of freedom
Multiple R-squared:  0.09763,   Adjusted R-squared:  0.09751
F-statistic: 839.2 on 15 and 116352 DF,  p-value: < 2.2e-16
```

- **R^2 múltiple**: 0.151 → El modelo explica solo el 15.1% de la variabilidad en la precipitación.
- **R^2 ajustado**: 0.1507 → Muy similar, lo que indica que el número de predictores no está inflando artificialmente el ajuste.
- **Error estándar residual**: 6.443 → Promedio de desviación entre valores observados y predichos.
- **F-statistic**: 552.2 con p-valor < 2.2e-16 → El modelo global es significativo.

Aunque el modelo es estadísticamente significativo, su capacidad explicativa es baja, lo que sugiere que la precipitación es difícil de predecir con estas variables o que hay no linealidades/interacciones no capturadas.

Variable	Coeficiente	p-valor	¿Significativo?	Justificación
(Intercept)	109.682	<2e-16	Sí	p < 0.001, altamente significativo
MinTemp	-0.063	3.50e-05	Sí	p < 0.001, efecto claro
MaxTemp	-0.393	<2e-16	Sí	p < 0.001, muy fuerte
Evaporation	+0.033	0.00571	Sí	p < 0.01, moderadamente significativo
Sunshine	-0.233	<2e-16	Sí	p < 0.001, muy fuerte

Variable	Coeficiente	p-valor	¿Significativo?	Justificación
WindGustSpeed	+0.072	<2e-16	Sí	$p < 0.001$, muy fuerte
WindSpeed9am	+0.034	4.43e-13	Sí	$p < 0.001$, muy fuerte
WindSpeed3pm	-0.070	<2e-16	Sí	$p < 0.001$, muy fuerte
Humidity9am	+0.092	<2e-16	Sí	$p < 0.001$, muy fuerte
Humidity3pm	+0.038	<2e-16	Sí	$p < 0.001$, muy fuerte
Pressure9am	-0.403	<2e-16	Sí	$p < 0.001$, muy fuerte
Pressure3pm	+0.268	<2e-16	Sí	$p < 0.001$, muy fuerte
Cloud9am	-0.013	0.43248	No	$p > 0.05$, no hay evidencia de efecto
Cloud3pm	-0.105	9.30e-10	Sí	$p < 0.001$, muy fuerte
Temp9am	+0.141	3.13e-10	Sí	$p < 0.001$, muy fuerte
Temp3pm	+0.458	<2e-16	Sí	$p < 0.001$, muy fuerte

- **Cloud9am** es la única variable no significativa ($p = 0.432$), lo que sugiere que no tiene un efecto estadísticamente detectable sobre la lluvia en este modelo.
- Todas las demás variables tienen p-valores muy bajos, lo que indica que sus efectos son estadísticamente significativos.
- La significancia estadística no implica causalidad ni relevancia práctica. Por ejemplo, aunque *Evaporation* tiene un efecto pequeño, es estadísticamente significativo.

4.5.5 Recrear la recta de la regresión lineal

Rainfall=109.682-0.063*MinTemp-0.393*MaxTemp+0.033*Evaporation-0.233*Sunshine+0.072*WindGustSpeed+
0.070*WindSpeed3pm+0.092*Humidity9am+0.038*Humidity3pm-0.403*Pressure9am+0.268*Pressure3pm-
0.013*Cloud9am-0.105*Cloud3pm+0.141*Temp9am+0.458*Temp3pm

- Si la temperatura mínima sube 1°C, la lluvia **disminuye 0.063 mm**. → No es un efecto grande, pero indica que noches más cálidas se asocian con menos lluvia.

- Si la temperatura máxima sube 1°C, la lluvia **baja 0.393 mm**. → Días más calurosos suelen ser más secos.
- Si la evaporación aumenta 1 mm, la lluvia **sube 0.033 mm**. → Sugiere que más humedad acumulada en el ambiente favorece la lluvia.
- Cada hora adicional de sol reduce la lluvia en **0.233 mm**. → Más sol = menos lluvia, relación muy intuitiva.
- Si las ráfagas aumentan 1 km/h, la lluvia **sube 0.072 mm**. → Las tormentas suelen venir acompañadas de ráfagas fuertes.
- Más viento por la mañana → **más lluvia**.
- Más viento por la tarde → **menos lluvia**. → Indica un patrón horario distinto en la dinámica del viento.
- Si la humedad matutina sube 1%, la lluvia aumenta **0.092 mm**. → Mucha humedad por la mañana es un buen predictor de lluvia.
- La humedad vespertina también aumenta la lluvia, pero menos que la matutina.
- Si la presión baja por la mañana, la lluvia **aumenta**. → La presión baja está asociada a sistemas inestables y lluvias.
- Si la temperatura a las 9am sube 1 grado, la lluvia aumenta **0.141 mm**. → Temperaturas matutinas más altas pueden indicar más humedad disponible.
- Si la temperatura a las 3pm sube 1 grado → **0.458 mm más de lluvia**.
- Así sería con todas las columnas

4.5.6 Predecir en el dataset de entrenamiento

```
predicciones <- predict(modelo, newdata = test)
predicciones[1]
```

```
        3
0.2846823
```

4.5.7 Evaluación de la regresión lineal

4.5.7.1 Error cuadrático medio (MSE)

```
mse <- mean((test$Rainfall - predicciones)^2, na.rm = TRUE)
mse
```

```
[1] 61.29287
```

4.5.7.2 Raíz del error cuadrático medio (RMSE)

```
rmse <- sqrt(mse)
rmse
```

```
[1] 7.828977
```

4.5.8 Error absoluto medio (MAE)

```
mae <- mean(abs(test$Rainfall - predicciones), na.rm = TRUE)
mae
```

```
[1] 3.430807
```

Comparamos con los estadísticos de la columna que hemos querido predecir, en este caso Rainfall:

```
summary(datos$Rainfall)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   0.000   0.000   2.308   0.600 371.000
```

Los errores son superiores a la media, por lo que el modelo está cometiendo errores **tan grandes o mayores** que el valor promedio de la variable. Esto implica que el modelo no está capturando bien la estructura del fenómeno.

4.5.9 R^2

```
summary(modelo)$r.squared
```

```
[1] 0.09762994
```

Un **R² de 0.0976** implica que el modelo explica **solo el 9.76% de la variabilidad** de la precipitación en el conjunto de test.

En otras palabras:

- El **90% de la variabilidad de la lluvia NO está siendo explicada** por las variables del modelo.
- Esto confirma que **la lluvia es extremadamente difícil de predecir** con un modelo lineal simple usando variables meteorológicas básicas.

4.6 Regresión lineal con Logaritmos

Se utiliza para que los cambios en vez de ser expresados en unidades se expresan en porcentajes

```
#Añadimos el +1 para que no haya infinitos ni vacíos -> se llama ajuste
modelo_log <- lm(log(Rainfall+1) ~ MinTemp + MaxTemp + Evaporation + Sunshine +
                 WindGustSpeed + WindSpeed9am + WindSpeed3pm + Humidity9am +
                 Humidity3pm + Pressure9am + Pressure3pm + Cloud9am + Cloud3pm +
                 Temp9am + Temp3pm, data = train)
#Resumen y análisis de la regresión logística
summary(modelo_log)
```

```
Call:
lm(formula = log(Rainfall + 1) ~ MinTemp + MaxTemp + Evaporation +
    Sunshine + WindGustSpeed + WindSpeed9am + WindSpeed3pm +
    Humidity9am + Humidity3pm + Pressure9am + Pressure3pm + Cloud9am +
    Cloud3pm + Temp9am + Temp3pm, data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-1.6853 -0.4963 -0.1992  0.2235  5.5422
```

```
Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)   -2.257e-01  1.692e-02 -13.339  < 2e-16 ***
MinTemp        4.900e-02  9.131e-04  53.660  < 2e-16 ***
MaxTemp       -2.175e-02  9.900e-04 -21.969  < 2e-16 ***
Evaporation   -6.261e-03  7.230e-04  -8.659  < 2e-16 ***
Sunshine      -5.316e-03  6.044e-04  -8.794  < 2e-16 ***
WindGustSpeed  7.765e-03  1.888e-04  41.139  < 2e-16 ***
WindSpeed9am   6.196e-03  3.345e-04  18.522  < 2e-16 ***
WindSpeed3pm  -4.461e-03  3.450e-04 -12.932  < 2e-16 ***
Humidity9am    9.588e-03  1.613e-04  59.452  < 2e-16 ***
Humidity3pm    2.590e-03  1.695e-04  15.278  < 2e-16 ***
Pressure9am   -2.517e-04  4.081e-05  -6.166 7.04e-10 ***
Pressure3pm    1.363e-04  4.096e-05   3.328 0.000876 ***
Cloud9am       1.598e-02  1.103e-03  14.483  < 2e-16 ***
Cloud3pm       9.570e-03  1.117e-03   8.568  < 2e-16 ***
Temp9am       -1.010e-02  1.152e-03  -8.767  < 2e-16 ***
Temp3pm       -1.009e-02  6.716e-04 -15.025  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7916 on 116352 degrees of freedom
Multiple R-squared:  0.2273,    Adjusted R-squared:  0.2272
F-statistic:  2282 on 15 and 116352 DF,  p-value: < 2.2e-16
```

Obtendremos los coeficientes en porcentajes de la siguiente forma:

```r
coef_porcentajes <- (exp(coef(modelo_log)) - 1) * 100
coef_porcentajes
```

```
  (Intercept)       MinTemp       MaxTemp   Evaporation      Sunshine
 -20.20564090    5.02154517   -2.15148994   -0.62412082   -0.53014749
WindGustSpeed  WindSpeed9am  WindSpeed3pm   Humidity9am   Humidity3pm
   0.77955340    0.62152639   -0.44513143    0.96336638    0.25929706
  Pressure9am   Pressure3pm      Cloud9am      Cloud3pm       Temp9am
  -0.02516199    0.01363233    1.61056373    0.96160826   -1.00487139
     Temp3pm
  -1.00401028
```

Variable	% cambio	Interpretación
MinTemp	**+5.02%**	Si la temperatura mínima sube 1°C, la lluvia aumenta un 5%.
MaxTemp	**–2.15%**	Si la temperatura máxima sube 1°C, la lluvia baja un 2.15%.
Temp9am	**–1.00%**	Más temperatura por la mañana → menos lluvia.
Temp3pm	**–1.00%**	Más temperatura por la tarde → menos

5 Aprendizaje Automático II (Clasificación)

Es una técnica que sirve para **predecir categorías**, como "sí/no" o "spam/no spam". Se usa cuando queremos **asignar cada caso a un grupo** según sus características.

5.1 Regresión Logística

Primero procedemos a la carga de datos

```
datos <- read.csv("C:\\Users\\garci\\Downloads\\weatherAUS.csv")
datos <- na.omit(datos)
datos$Llueve_Hoy <- ifelse(datos$RainToday == "Yes", 1, 0)
View(datos)
```

Seleccionamos columnas (en el caso de tener columnas que son de tipo texto primero se deberían pasar a factores o utilizar conversiones con dummies.

```
library(dplyr)
```

```
Adjuntando el paquete: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
datos_regresion <- datos%>%select("MinTemp","MaxTemp","Rainfall","Evaporation","Sunshine","W
```

5.1.1 Dividir el dataset en entrenamiento y testeo

```
set.seed(123) # para reproducibilidad y aleatoriedad
# Proporción de entrenamiento (80%). Seleccionamos un 80% de los datos para el entrenamiento
indice <- sample(1:nrow(datos_regresion), size = 0.8 * nrow(datos_regresion))
train <- datos_regresion[indice, ] # datos de entrenamiento
test <- datos_regresion[-indice, ] # datos de testeo
```

5.1.2 Entrenar el modelo de regresión logística

```
modelo <- glm( Llueve_Hoy ~ MinTemp + MaxTemp +Rainfall+ Evaporation + Sunshine + WindGustSp
```

```
Warning: glm.fit: algorithm did not converge
```

```
Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

5.1.2.1 Análisis del modelo

```
summary(modelo)
```

```
Call:
glm(formula = Llueve_Hoy ~ MinTemp + MaxTemp + Rainfall + Evaporation +
    Sunshine + WindGustSpeed + WindSpeed9am + WindSpeed3pm +
    Humidity9am + Humidity3pm + Pressure9am + Pressure3pm + Cloud9am +
    Cloud3pm + Temp9am + Temp3pm, family = "binomial", data = train)

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)   -2.592e+02  1.151e+04  -0.023    0.982
MinTemp       -7.458e-01  3.525e+01  -0.021    0.983
MaxTemp       -1.010e+00  9.049e+01  -0.011    0.991
Rainfall       1.937e+02  8.428e+02   0.230    0.818
Evaporation   -6.903e-02  1.858e+01  -0.004    0.997
Sunshine      -2.628e-01  4.405e+01  -0.006    0.995
WindGustSpeed  6.622e-03  1.024e+01   0.001    0.999
```

```
WindSpeed9am   3.390e-02  1.398e+01   0.002    0.998
WindSpeed3pm  -7.408e-02  1.338e+01  -0.006    0.996
Humidity9am    6.504e-02  8.588e+00   0.008    0.994
Humidity3pm   -1.086e-01  1.147e+01  -0.009    0.992
Pressure9am    3.999e-01  4.848e+01   0.008    0.993
Pressure3pm   -3.504e-01  5.004e+01  -0.007    0.994
Cloud9am       1.589e-02  5.325e+01   0.000    1.000
Cloud3pm       5.045e-01  6.113e+01   0.008    0.993
Temp9am        1.130e+00  6.026e+01   0.019    0.985
Temp3pm        4.867e-01  1.051e+02   0.005    0.996

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 4.7633e+04  on 45135  degrees of freedom
Residual deviance: 2.3913e-04  on 45119  degrees of freedom
AIC: 34

Number of Fisher Scoring iterations: 25
```

Nota.- Cuando en R os aparece un número seguido de un e+05 es notación científica quiere decir $*10^5$ de la misma forma e+02 sería $*10^2$ y e-03 sería $*10^{-3}$.

Aunque en este curso no profundizamos en si se estima bien o no los coeficientes, lo podrías ver ya que hay coeficientes que son muy grandes pero sus errores son muy muy elevados también, por ejemplo,

- Rainfall = +193.7 -> error SE = 842.8
- Temp3pm = +0.4867 -> error SE=105.1
- Cloud9am = +0.01589 -> error SE= 53.25

Esto indica que el modelo no está estimando bien los coeficientes. Probablemente hay:

- Multicolinealidad extrema
- Datos mal escalados o con outliers
- Variables redundantes o correlacionadas

5.1.2.1.1 Significancia

Todos los valores de Pr(|z|) son altísimos (0.99) y mayores que 0.05, por lo que ninguna variable se considera signficativa.

5.1.2.1.2 Coeficientes

En la regresión logística, la magnitud no se interpreta igual que en la regresión lineal:

- Interpretamos el signo. Si es negativo implica una relación inversa y si es positivo entonces implica una relación positiva (a más de la columna, menos probabilidades de la variable que clasifico)
- Interpretamos los valores. Para interpretar los coeficientes se hace exp(valor_estimado_coeficiente), por ejemplo en el caso de WindSpeed9am que es 3.390e-02=0.0339 sería:

```
exp(0.0339)
```

```
[1] 1.034481
```

```
exp(3.390e-02)
```

```
[1] 1.034481
```

Restamos 1 y multiplicamos por 100:

```
(exp(0.0339)-1)*100
```

```
[1] 3.448115
```

Es decir, cada vez que aumenta en 1 la variable WindSpeed9am la probabilidad de que llueva hoy aumenta 3.448%

5.1.3 Realizar las predicciones y evaluar el resultado

```
# Obtener probabilidades predichas
prob_pred <- predict(modelo, newdata = test, type = "response")
clase_pred <- ifelse(prob_pred > 0.5, 1, 0)
# Matriz de confusión
table(Real = test$Llueve_Hoy, Predicho = clase_pred)
```

```
    Predicho
Real    0    1
   0 8779    0
   1    0 2505
```

5.1.4 Matriz de confusión

```
library(caret)
```

```
Cargando paquete requerido: ggplot2
```

```
Cargando paquete requerido: lattice
```

```
pred_clase_factor <- factor(clase_pred, levels = c(0,1))
real_factor <- factor(test$Llueve_Hoy, levels = c(0,1))

confusionMatrix(pred_clase_factor, real_factor)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 8779    0
         1    0 2505

               Accuracy : 1
                 95% CI : (0.9997, 1)
    No Information Rate : 0.778
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.000
            Specificity : 1.000
         Pos Pred Value : 1.000
         Neg Pred Value : 1.000
             Prevalence : 0.778
         Detection Rate : 0.778
   Detection Prevalence : 0.778
      Balanced Accuracy : 1.000

       'Positive' Class : 0
```

5.1.4.1 Análisis de VP, VN, FP y FN

El *positive class* es **0**, es decir:

- **0 = "No llueve hoy"** (clase mayoritaria)
- **1 = "Llueve hoy"**

Destacamos:

- VP: Verdadero Positivo. La predicción dice que es positivo y en realidad es positivo: 8751
- FP: Falso Positivo. La predicción nos dice que es positivo (en este caso 0) y en realidad es negativo (1): 1438
- FN: Falso Negativo. La predicción dice que es negativo (en este caso 1) y en realidad es positivo (en este caso 0): 28
- VN: Verdadero Negativo. La predicción dice que es negativo y en realidad es negativo: 1067

Por lo tanto se puede analizar:

- El modelo **acierta muchísimo** cuando dice que NO llueve (8751 aciertos).
- El modelo **falla bastante** cuando dice que NO llueve pero sí llueve (1438 falsos negativos).
- El modelo **casi nunca** predice lluvia cuando no la hay (solo 28 falsos positivos).

Esto es importante para interpretar sensibilidad y especificidad.

5.1.4.2 Accuracy

El accuracy es 0.8701, es decir, un 87% de acierto que es muy elevado.

5.1.4.3 Sensitividad o la sensibilidad (para la clase 0)

Es la capacidad del modelo para acertar cuando el caso realmente pertenece a la clase positiva. En otras palabras: mide cuántos de los casos positivos detecta correctamente.

La sensitividad es 0.9969, por lo que el modelo detecta casi todos los días sin lluvia.

5.1.4.4 Especificidad (para la clase 1)

Es la capacidad del modelo para acertar cuando el caso realmente pertenece a la clase negativa. Es decir: mide cuántos de los casos negativos identifica correctamente.

La especificidad es 0.4259, el modelo detecta el 42.59% de los días con lluvia. Es decir, más de la mitad de los días con lluvia los clasifica como "no llueve".

5.1.5 Predecir la probabilidad de un caso concreto

La Oficina Meteorológica ha entrenado un modelo de regresión logística para predecir si **llueve hoy** a partir de distintas variables climáticas registradas el día anterior y durante la mañana. El modelo utiliza información sobre temperaturas, viento, humedad, presión atmosférica, nubosidad y lluvia previa.

Para un día concreto, se han registrado los siguientes valores:

La **temperatura mínima del día anterior** fue de **12.3 °C**, mientras que la **temperatura máxima** alcanzó **21.8 °C**. Durante ese día cayeron **4.2 mm de lluvia**, y la **evaporación** medida fue de **3.1 mm**. Además, el día anterior hubo **6.5 horas de sol**.

En cuanto al viento, la **racha máxima registrada** alcanzó los **48 km/h**. A las **9 de la mañana**, la velocidad del viento era de **19 km/h**, y a las **3 de la tarde** aumentó hasta **24 km/h**.

La **humedad relativa** era del **82% a las 9 am** y del **67% a las 3 pm**. La **presión atmosférica** fue de **1012.4 hPa a las 9 am** y descendió ligeramente hasta **1010.8 hPa a las 3 pm**.

Respecto a la nubosidad, a las **9 am** se observaron **6 octas** (de un máximo de 8), y a las **3 pm** la nubosidad aumentó a **7 octas**. Finalmente, la **temperatura a las 9 am** era de **15.4 °C**, y a las **3 pm** subió hasta **20.1 °C**

```
probabilidad_lluvia <- predict(
  modelo,
  newdata = data.frame(
    MinTemp = 12.3,
    MaxTemp = 21.8,
    Rainfall = 4.2,
    Evaporation = 3.1,
    Sunshine = 6.5,
    WindGustSpeed = 48,
    WindSpeed9am = 19,
    WindSpeed3pm = 24,
    Humidity9am = 82,
```

```
    Humidity3pm = 67,
    Pressure9am = 1012.4,
    Pressure3pm = 1010.8,
    Cloud9am = 6,
    Cloud3pm = 7,
    Temp9am = 15.4,
    Temp3pm = 20.1
  ),
  type = "response"
)

cat(probabilidad_lluvia)
```

```
1
```

En este caso saldría que la probabilidad de llover es del 1 por lo que hay un 100% de probabilidades de llover.

```
valor_predicho <- ifelse(probabilidad_lluvia>0.5, 1, 0)
valor_predicho
```

```
1
1
```

El valor predicho es 1, es decir, llueve hoy.

5.2 Árbol de decisión

Los árboles de decisión sirven para hacer también clasificaciones pero a diferencia de la regresión logística que solo puede predecir clases binarias (0 o 1, Sí o No), en el caso del árbol de decisión se puede clasificar en dos o más categorías.

```
library(rpart) #Para realizar el árbol
library(rpart.plot) #Para dibujar el árbol
```

La sintaxis es muy parecida a la de regresión logística solo que poniendo rpart en vez de glm:

```
modeloarbol <- rpart( Llueve_Hoy ~ MinTemp + MaxTemp +Rainfall+ Evaporation + Sunshine + Wind
```

5.2.1 Interpretación

Veamos el resumen, aunque **lo importante del modelo de árbol es dibujarlo**

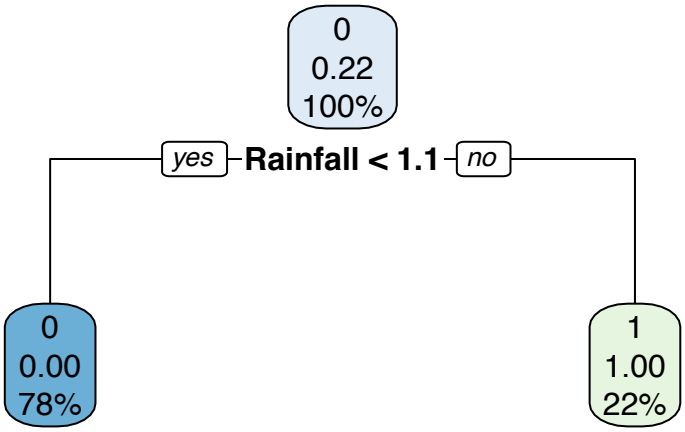
```
summary(modeloarbol)
```

```
Call:
rpart(formula = Llueve_Hoy ~ MinTemp + MaxTemp + Rainfall + Evaporation +
    Sunshine + WindGustSpeed + WindSpeed9am + WindSpeed3pm +
    Humidity9am + Humidity3pm + Pressure9am + Pressure3pm + Cloud9am +
    Cloud3pm + Temp9am + Temp3pm, data = train, method = "class")
  n= 45136

    CP nsplit rel error xerror        xstd
1 1.00      0         1      1 0.008847406
2 0.01      1         0      0 0.000000000

Variable importance
   Rainfall Humidity9am Humidity3pm    Sunshine Pressure9am     Temp3pm
         85           6           5           2           1           1

Node number 1: 45136 observations,    complexity param=1
  predicted class=0  expected loss=0.2206  P(node) =1
    class counts: 35179  9957
   probabilities: 0.779 0.221
  left son=2 (35179 obs) right son=3 (9957 obs)
  Primary splits:
      Rainfall    < 1.05    to the left,  improve=15520.970, (0 missing)
      Humidity9am < 76.5    to the left,  improve= 2063.716, (0 missing)
      Humidity3pm < 59.5    to the left,  improve= 1784.713, (0 missing)
      Sunshine    < 8.45    to the right, improve= 1479.595, (0 missing)
      Cloud9am    < 3.5     to the left,  improve= 1034.508, (0 missing)
  Surrogate splits:
      Humidity9am < 89.5    to the left,  agree=0.796, adj=0.075, (0 split)
      Humidity3pm < 77.5    to the left,  agree=0.792, adj=0.057, (0 split)
      Sunshine    < 0.05    to the right, agree=0.783, adj=0.018, (0 split)
      Pressure9am < 1002.35 to the right, agree=0.782, adj=0.013, (0 split)
      Temp3pm     < 10.25   to the right, agree=0.782, adj=0.010, (0 split)
```

```
Node number 2: 35179 observations
  predicted class=0  expected loss=0  P(node) =0.7794
    class counts: 35179     0
   probabilities: 1.000 0.000

Node number 3: 9957 observations
  predicted class=1  expected loss=0  P(node) =0.2206
    class counts:     0  9957
   probabilities: 0.000 1.000
```

El summary nos da la misma información casi que el dibujo pero es más interpretable verlo dibujado:

```
rpart.plot(modeloarbol)
```



En primer lugar los árboles están compuestos por nodos que serían los tres óvalos dos azules y uno verde que nos ha dado el modelo. En cada nodo, está compuesto por tres números:

- El número de arriba, nos indica si hay más 0s o 1s
- El número del medio, nos indica el porcentaje de 1s que hay en la muestra
- El número de abajo, el porcentaje **sobre el total de la muestra**

Por ejemplo el nodo azul claro vemos que lo que más hay son 0, además hay un 0.22 de probabilidad de 1 y en este nodo se ha analizado el 100% de la muestra.

El nodo azul oscuro recoge aquellos casos en que Rainfall es menor que 1.1. En este nodo se encuentra el 78% de los casos, lo que más hay son 0 y la probabilidad de 1 es 0.

El nodo verde claro recoge aquellos casos en que Rainfall NO es menor que 1.1. En este nodo se encuentra el 22% de los datos, donde lo que más hay son 1 y la probabilidad de 1 es 1.

5.2.2 Importancia de las variables

Uno de los aspectos más importantes es conocer cuál ha sido la importancia de las variables, esto se puede hacer con:

```
modeloarbol$variable.importance
```

```
   Rainfall Humidity9am Humidity3pm    Sunshine Pressure9am     Temp3pm
 15520.9723   1159.7473    882.2808    283.7016    204.2028    155.8800
```

Esto nos refleja la improtancia de las vairables, donde Rainfall es la variable con mayor importancia y Temp3pm es la que menos importancia tiene.

5.2.3 Predicción

```
prediccionarbol <- predict(modeloarbol,type = "class", newdata=test)
prediccionarbol
```

```
  6050   6054   6063   6064   6065   6070   6072   6073   6074   6085   6086
     0      0      0      0      0      0      1      1      0      0      0
  6087   6090   6101   6112   6115   6116   6117   6118   6121   6126   6140
     0      0      0      0      0      0      0      0      1      0      0
  6141   6149   6157   6158   6160   6164   6168   6178   6188   6191   6194
     0      1      0      0      0      0      0      0      1      0      0
  6195   6196   6197   6204   6206   6208   6212   6217   6218   6221   6225
     0      1      0      0      0      0      0      0      0      0      0
  6230   6235   6241   6245   6254   6255   6258   6263   6271   6290   6294
     0      0      0      1      0      0      0      0      0      0      0
  6300   6305   6306   6309   6310   6311   6313   6330   6333   6338   6343
     0      0      0      0      1      0      0      0      0      0      0
  6352   6354   6359   6365   6367   6368   6369   6370   6380   6388   6390
```

```
141823 141824 141832 141833 141834 141837 141849 141850 141851 141858 141867
     0      0      0      1      1      1      0      0      0      0      1
141879 141882 141887 141888 141892 141893 141904 141908 141909 141911 141920
     0      0      1      1      0      0      0      0      0      0      0
141926 141934 141951 141966 141970 141982 141985 141990 141997 141998 142002
     0      0      0      0      0      0      0      0      0      0      0
142007 142008 142009 142010 142014 142016 142043 142053 142062 142065 142067
     0      0      1      0      0      1      0      0      1      0      0
142070 142071 142072 142080 142081 142087 142096 142099 142107 142113 142120
     0      0      0      0      0      1      0      0      0      1      1
142123 142134 142135 142137 142142 142145 142153 142155 142160 142172 142173
     0      1      0      0      0      1      1      0      1      1      1
142179 142180 142181 142188 142191 142193 142196 142198 142204 142209 142215
     1      0      0      0      1      1      1      0      1      1      0
142218 142219 142221 142224 142235 142245 142247 142253 142259 142261 142267
     1      1      0      1      1      0      0      0      0      0      0
142268 142272 142277 142281 142284 142285 142287 142289 142301
     0      0      0      0      0      0      0      0      0
Levels: 0 1
```

El número de arriba sería la fila y el de abajo el valor predicho si es 0 o 1 en este caso.

De la misma forma podemos predecir para un caso concreto:

```r
lluvia <- predict(
  modeloarbol,
  newdata = data.frame(
    MinTemp = 12.3,
    MaxTemp = 21.8,
    Rainfall = 4.2,
    Evaporation = 3.1,
    Sunshine = 6.5,
    WindGustSpeed = 48,
    WindSpeed9am = 19,
    WindSpeed3pm = 24,
    Humidity9am = 82,
    Humidity3pm = 67,
    Pressure9am = 1012.4,
    Pressure3pm = 1010.8,
    Cloud9am = 6,
    Cloud3pm = 7,
    Temp9am = 15.4,
    Temp3pm = 20.1
```

```
  ),
  type = "prob"   # rpart usa "prob" para probabilidades
)

lluvia
```

```
  0 1
1 0 1
```

- Primera fila sería las clases, es decir clase 0 y clase 1. La fila encabezada por 1 (que es el número de fila) son las probabilidades asociadas, es decir que se puede interpretar en este caso concreto como:
  - Probabilidad de clase **0** = **0**
  - Probabilidad de clase **1** = **1**

NOTA.- En clases hemos visto la regresión logística y el árbol de decisión con los datos de titanic que resulta en el que obtenemos un mejor resultado.

6 Clusterización

La clusterización consiste en descubrir patrones naturales dentro de un conjunto de datos, formando grupos (clusters) donde:

- Los elementos dentro del mismo grupo son similares
- Los elementos de grupos distintos son diferentes.

La clusterización se encuentra dentro del **aprendizaje no supervisado** puesto que no tenemos datos etiquetados, en otras palabras no partimos de un histórico ni unas categorías en las cuales se deben agrupar.

En un lenguaje más coloquial lo podemos entender como que el ordenador hace grupos de datos en base a lo que él define como similaridad.

6.1 Carga de Datos

En este caso no utilizaremos un archivo excel o csv, pero con ellos sería igual:

```
data("iris")
datos <- iris
```

6.2 Análisis Previo de los Datos

```
str(datos)
```

```
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Tenemos filas y columnas de las cuales cuatro son numéricas y la quinta son factores es decir es una columna cuyos valores son texto pero solo pueden ser tres opciones: setosa, versicolor o virginica.

Es muy importante que entendamos que la clusterización no está hecho para recrear la clasificación de Species (en todo caso si pretendieramos esto tendríamso que hacer una clasificación que forma parte del aprendizaje supervisado), si no que la idea es dar al ordenador las cuatro primeras columnas y que sea el ordenador el que haga los grupos como quiera.

Es decir, **no tiene por qué coincidir los clusters de la clusterización con clasificaciones prexistentes de los datos.**

Tenemos que preguntarnos si **existen valores nulos?**

```
colSums(is.na(datos))
```

```
Sepal.Length  Sepal.Width Petal.Length  Petal.Width      Species
           0            0            0            0            0
```

Vemos que no hay datos nulos, en caso de que los hubiera tendríamos que decidir si:

- Eliminar la columna
- Eliminar las filas con nulos
- Reemplazar los nulos con otro valor

6.3 Seleccionar las variables numéricas

Para utilziar el algoritmo de Kmeans solo pueden haber variables numéricas, por lo que el resto habría que convertirlo en variables numéricas.

```
library(dplyr)
```

```
Adjuntando el paquete: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag
```

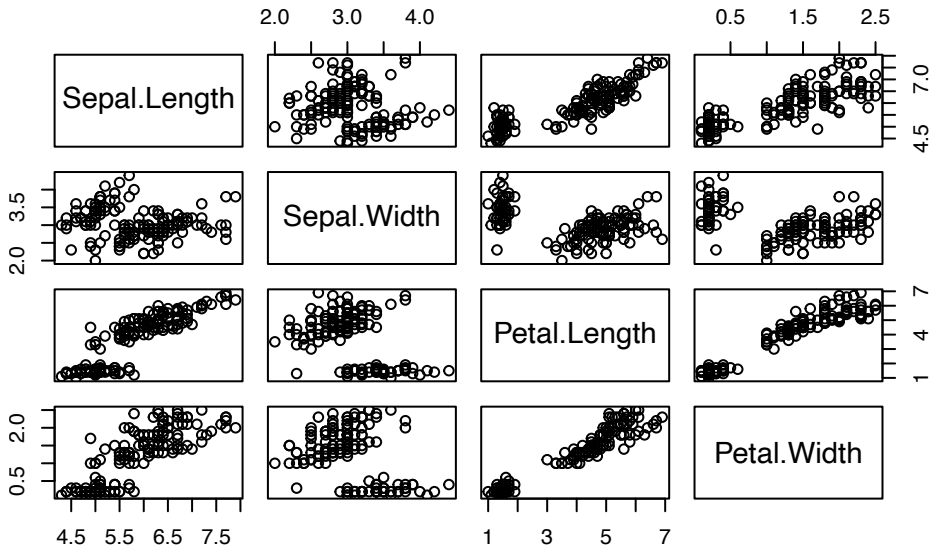
```
The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
datos_num<-datos%>%select(where(is.numeric))
```

6.4 Representar los datos

Podemos representar los datos mediante una matriz de gráficas

```
plot(datos_num)
```



Tenemos que:

- Fila 1 -> Longitud del Sépalo
- Fila 2 -> Ancho del sépalo
- Fila 3 -> Longitud del pétalo
- Fila 4-> Ancho del pétalo
- Columna 1 -> Longitud del Sépalo

- Columna 2 -> Ancho del sépalo
- Columna 3 -> Longitud del pétalo
- Columna 4-> Ancho del pétalo

Por lo tanto la fila 3 columna 4 es la representación de la longitud del pétalo vs el ancho del pétalo. Si nos damos cuenta se aprecia que la nube de puntos tiene un marcado caracter ascendente y se aproxima a una recta -> correlación positiva fuerte. Veamos el valor de las correlaciones

```
cor(datos_num)
```

```
             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length    1.0000000  -0.1175698    0.8717538   0.8179411
Sepal.Width    -0.1175698   1.0000000   -0.4284401  -0.3661259
Petal.Length    0.8717538  -0.4284401    1.0000000   0.9628654
Petal.Width     0.8179411  -0.3661259    0.9628654   1.0000000
```

Podemos ver lo siguiente (tendríamos que explicarlo con más detalle):

- Correlaciones positivas fuertes (mayores de 0.5):
  - Longitud del sépalo y longitud del pétalo
  - Longitud del sépalo y ancho del pétalo
  - Longitud del pétalo y ancho del pétalo
- Correlaciones negativas débiles (entre 0 y -0.5):
  - Ancho del sépalo y longitud del pétalo
  - Ancho del sépalo y ancho del pétalo
  - Ancho del sépalo y longitud del sépalo

Como podemos ver en efecto la longitud del pétalo y el ancho del pétalo tienen una fuerte correlación positiva.

6.5 Decidir cuál es el número de grupos óptimos

Podemos decidir cuál es el número de grupos óptimos de tres formas:

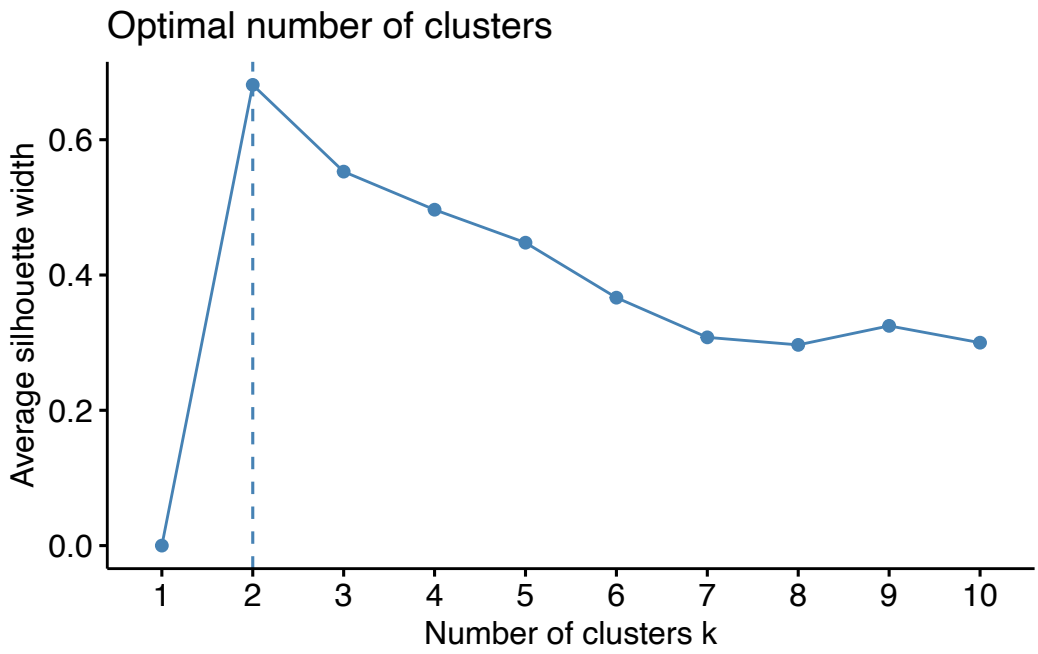
6.5.1 Librería factoextra por defecto

```
#install.packages("factoextra")
library(factoextra)
```

```
Cargando paquete requerido: ggplot2
```

```
Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
fviz_nbclust(datos_num,kmeans)
```

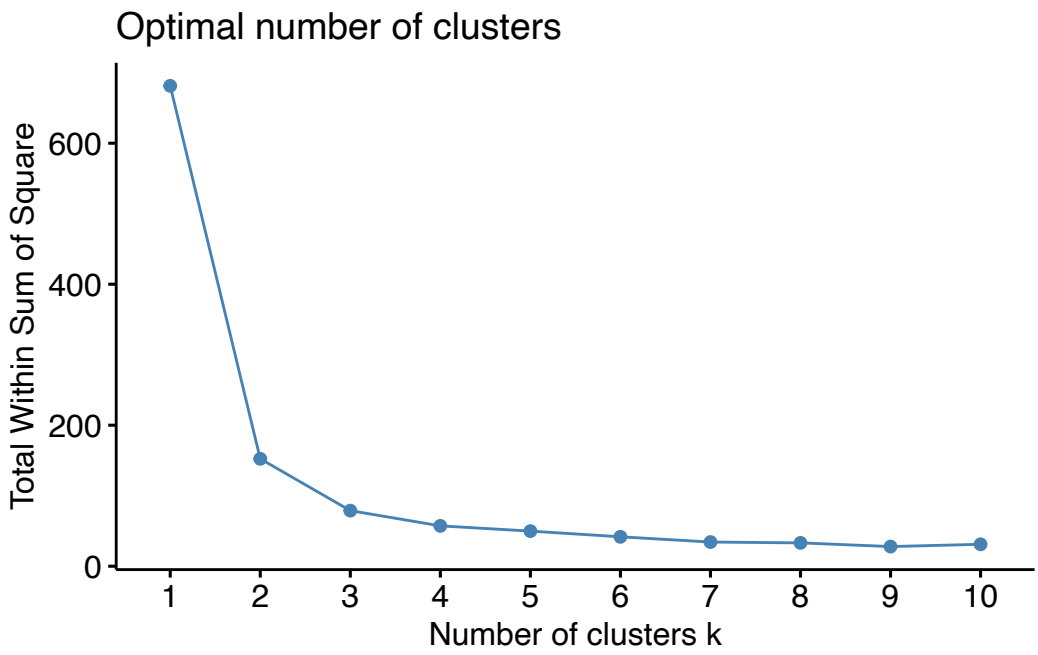


Si vemos la gráfica (a vosotros os aparecerá en la parte de plot del cuarto panel (esquina inferior derecha de RStudio) vemos que el valor recomendado es , ahora bien en este caso se está haciendo con el método por defecto de la función.

6.5.2 Método del codo

Este es el método más utilizado, aunque para algunos es algo manual, también se utiliza en otros lenguajes de programación como Python. Veamos como podemos utilizarlo:

```
fviz_nbclust(datos_num,kmeans, method = "wss")
```

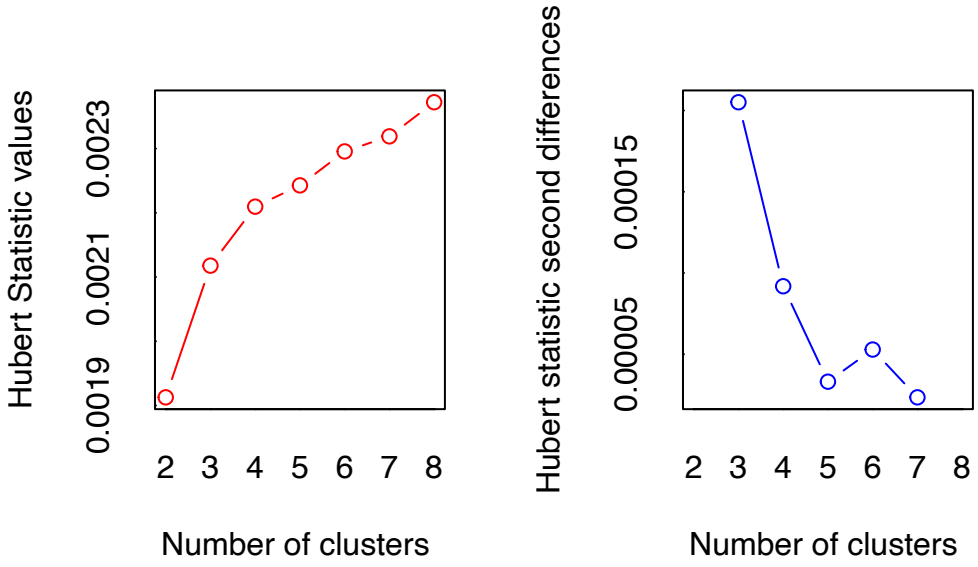


En el gráfico resultante, el número óptimo de clusteres, viene determiando por donde exista un "codo", en este caso, el 3 es el número óptimo ya que es donde, aparece "el codo". Algunos podríais haber considerado 2 y también sería correcto. Entre 2 y 3 se escoge el que mejor nos da la clusterización después.

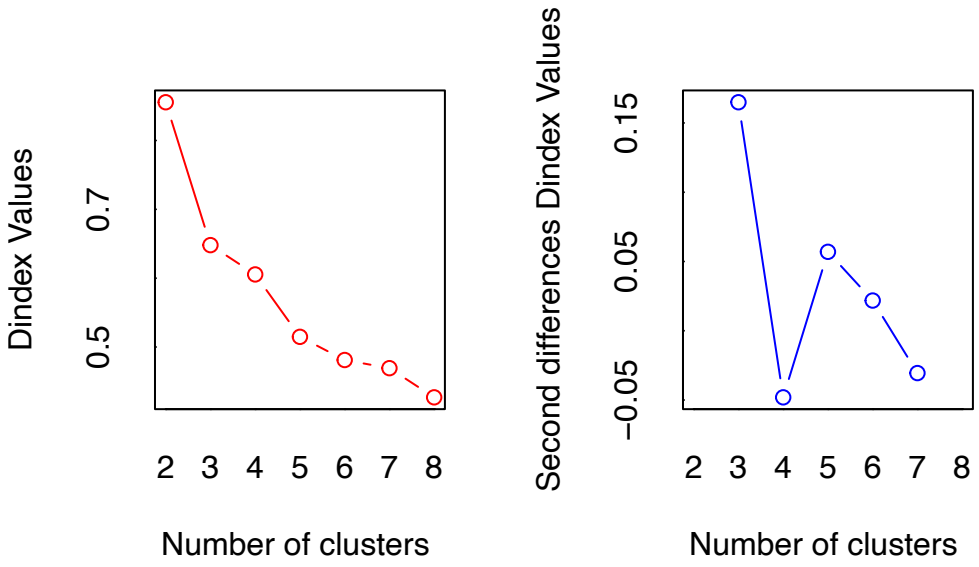
6.5.3 Método nbcluster

Para aquellos que el método de codo les parece muy manual existe el método nbcluster que utiliza varios criterios para establecer cual sería el número óptimo de clústers.

```
#install.packages("NbClust")
library(NbClust)
NbClust(datos_num,min.nc = 2,max.nc = 8, method = "kmeans")
```




```
*** : The Hubert index is a graphical method of determining the number of clusters.
                In the plot of Hubert index, we seek a significant knee that corresponds to a
                significant increase of the value of the measure i.e the significant peak in
                index second differences plot.
```




```
*** : The D index is a graphical method of determining the number of clusters.
                In the plot of D index, we seek a significant knee (the significant peak in
                second differences plot) that corresponds to a significant increase of the va
                the measure.

*******************************************************************
* Among all indices:
* 11 proposed 2 as the best number of clusters
* 11 proposed 3 as the best number of clusters
* 2 proposed 8 as the best number of clusters

                   ***** Conclusion *****

* According to the majority rule, the best number of clusters is  2


*******************************************************************

$All.index
       KL       CH Hartigan     CCC    Scott  Marriot    TrCovW   TraceW
2  5.9068 513.9245 137.9491 35.9428 1044.605 467371.6 1045.9696 152.3480
3 12.4890 561.6278  15.2384 37.6701 1246.668 273408.6  248.9814  78.8514
```

```
4  0.2006 415.4655  63.3643 33.4414 1296.511 348642.4  221.8632  71.4452
5  4.2008 459.5058  25.2963 34.3409 1415.662 246163.9  140.6758  49.8223
6  2.0868 433.7827   6.3415 33.3853 1505.840 194309.4   79.9311  42.4215
7  0.2423 375.8334  34.3200 31.4378 1523.492 235115.0   74.1212  40.6322
8  1.8336 401.5324  23.0039 32.0912 1607.479 175426.3   54.5852  32.7679
   Friedman    Rubin Cindex     DB Silhouette   Duda Pseudot2   Beale Ratkowsky
2  732.8086  62.6152 0.2728 0.4744     0.6810 1.9253 -52.8667 -1.1380    0.5462
3  801.6490 120.9780 0.3450 0.7256     0.5528 0.5112  45.9014  2.2615    0.4967
4  850.7642 133.5189 0.3485 0.9484     0.4152 0.5870  46.4403  1.6709    0.4440
5  936.2996 191.4664 0.3327 0.9987     0.3728 1.0842  -3.8046 -0.1814    0.4067
6 1070.5134 224.8690 0.3596 1.0808     0.3357 1.0328  -1.1759 -0.0739    0.3733
7 1086.0457 234.7718 0.3614 1.1374     0.2935 1.2705  -7.2384 -0.4949    0.3481
8 1298.5148 291.1170 0.3830 1.1072     0.3121 0.6969  10.4369  1.0042    0.3301
     Ball Ptbiserial     Frey McClain   Dunn Hubert SDindex Dindex   SDbw
2 76.1740     0.8345   1.7571  0.2723 0.0765 0.0019  1.2319 0.8556 0.1618
3 26.2838     0.7146 -20.7325  0.5255 0.0988 0.0021  1.6973 0.6480 0.2257
4 17.8613     0.6250   1.0780  0.7275 0.0528 0.0022  3.0992 0.6055 0.2351
5  9.9645     0.5521   1.3588  0.9903 0.0624 0.0022  3.1825 0.5148 0.1542
6  7.0703     0.5034 -11.9269  1.2043 0.0739 0.0023  3.3461 0.4811 0.1149
7  5.8046     0.4724   0.4676  1.3897 0.0739 0.0023  4.4801 0.4693 0.0903
8  4.0960     0.4566   0.5934  1.4577 0.0869 0.0024  4.6264 0.4269 0.0716

$All.CriticalValues
  CritValue_Duda CritValue_PseudoT2 Fvalue_Beale
2         0.5633            85.2756       1.0000
3         0.5551            38.4707       0.0640
4         0.5842            46.9673       0.1574
5         0.4837            52.3097       1.0000
6         0.4590            43.6093       1.0000
7         0.4590            40.0734       1.0000
8         0.4284            32.0211       0.4098

$Best.nc
                     KL       CH Hartigan     CCC    Scott  Marriot   TrCovW
Number_clusters  3.000   3.0000   3.0000  3.0000   3.0000      3.0   3.0000
Value_Index     12.489 561.6278 122.7107 37.6701 202.0631 269196.9 796.9882
                 TraceW Friedman   Rubin Cindex     DB Silhouette   Duda
Number_clusters  3.0000   8.0000   3.000 2.0000 2.0000      2.000 2.0000
Value_Index     66.0903 212.4691 -45.822 0.2728 0.4744      0.681 1.9253
                PseudoT2  Beale Ratkowsky    Ball PtBiserial   Frey McClain
Number_clusters   2.0000  2.000    2.0000  3.0000     2.0000 2.0000  2.0000
Value_Index     -52.8667 -1.138    0.5462 49.8902     0.8345 1.7571  0.2723
                  Dunn Hubert SDindex Dindex   SDbw
```

```
Number_clusters 3.0000      0  2.0000      0 8.0000
Value_Index     0.0988      0  1.2319      0 0.0716

$Best.partition
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [38] 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [75] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2
[112] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[149] 2 2
```

Entre los resultados que nos aparecen en la consola destaca:

```
*******************************************************************
* Among all indices:
* 11 proposed 2 as the best number of clusters
* 11 proposed 3 as the best number of clusters
* 2 proposed 8 as the best number of clusters
***** Conclusion *****
* According to the majority rule, the best number of clusters is  2      ********************
```

El propio algoritmo nos recomienda 2 aunque si vemos la misma cantidad de criterios también dijeron 3, por lo que estaríamos de nuevo entre 2 y 3.

6.6 Aplicar el algoritmo KMeans

```
clusterizacion <- kmeans(datos_num,2) #Seleccionamos 2 o 3 el que escojamos como número óptir
clusterizacion
```

```
K-means clustering with 2 clusters of sizes 97, 53

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     6.301031    2.886598     4.958763    1.695876
2     5.005660    3.369811     1.560377    0.290566

Clustering vector:
  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [38] 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1
```

```
[112] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[149] 1 1

Within cluster sum of squares by cluster:
[1] 123.79588  28.55208
 (between_SS / total_SS =  77.6 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

6.7 Interpretación del clustering

Vemos que el algoritmo KMeans nos ha agrupado los datos en dos clusters, uno con 97 elementos y otro con 53.

Cuando miramos las medias de los clústers podemos interpretar que características tienen cada uno de ellos:

- Clúster 1 tiene valores bastante mayores en la longitud del sépalo, longitud del pétalo y ancho del pétalo por lo que podríamos considerar que son flores grandes.
- Clúster 2 serían flores más pequeñas (a grosso modo)

Además, el clustering vector nos dice a qué grupo pertenece cada una de las flores aunque también se puede obtener como:

```
clusterizacion$cluster
```

```
  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [38] 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1
[112] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[149] 1 1
```

Podemos mirar la distribución en los clústers:

```
table(clusterizacion$cluster)
```

```
 1  2
97 53
```

El clúster 1 es el que tiene 97 flores y el clúster 2 tendría 53.

En este caso concreto y especial en el que tenemos otra columna con una clasificación previa podemos mirar como están relacionadas:

```
table(datos$Species, clusterizacion$cluster)
```

```
            1  2
  setosa    0 50
  versicolor 47  3
  virginica  50  0
```

Podemos comprobar que el clsuter 2 serían las setosa y alguna versicolor y el 1 correspondería a las virginica y la mayoría de versicolor. Que no coincidan está bien ya que en este caso buscamos una agrupación completamente diferente y no basada en la clasificación previa.

6.8 Interpretación de los clusters utilizando árboles

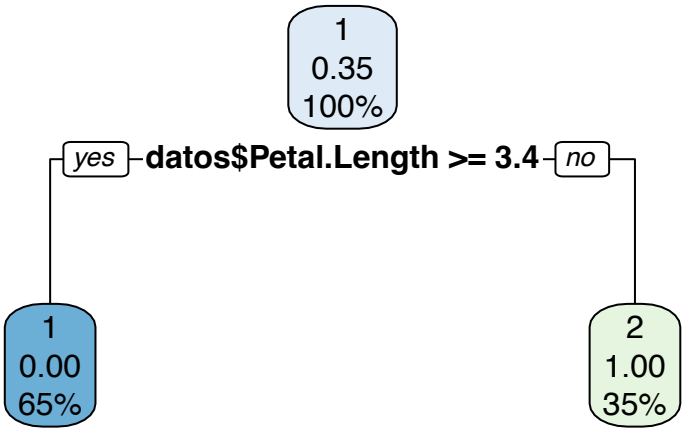
Una de las formas de entender algo mejor como ha realizado la clusterización es una vez hecha la misma realizar una clasificación con árboles de decisión sobre todo el conjunto de datos para intentar entender la lógica

```
library(rpart)
library(rpart.plot)
# ahora, podemos calcular el árbol de decisión
arbol <- rpart(clusterizacion$cluster~datos$Sepal.Length+
                                      datos$Sepal.Width+
                                      datos$Petal.Length+
                                      datos$Petal.Width,
                                      method="class")
summary(arbol)
```

```
Call:
rpart(formula = clusterizacion$cluster ~ datos$Sepal.Length +
    datos$Sepal.Width + datos$Petal.Length + datos$Petal.Width,
```

```
    method = "class")
  n= 150

    CP nsplit rel error xerror      xstd
1 1.00      0         1      1 0.1104593
2 0.01      1         0      0 0.0000000

Variable importance
datos$Petal.Length  datos$Petal.Width datos$Sepal.Length  datos$Sepal.Width
                31                 29                 26                 15

Node number 1: 150 observations,    complexity param=1
  predicted class=1  expected loss=0.3533333  P(node) =1
    class counts:    97    53
   probabilities: 0.647 0.353
  left son=2 (97 obs) right son=3 (53 obs)
  Primary splits:
      datos$Petal.Length < 3.4  to the right, improve=68.54667, (0 missing)
      datos$Petal.Width  < 0.8  to the right, improve=62.72667, (0 missing)
      datos$Sepal.Length < 5.45 to the right, improve=51.67226, (0 missing)
      datos$Sepal.Width  < 3.35 to the left,  improve=23.05898, (0 missing)
  Surrogate splits:
      datos$Petal.Width  < 0.8  to the right, agree=0.980, adj=0.943, (0 split)
      datos$Sepal.Length < 5.45 to the right, agree=0.940, adj=0.830, (0 split)
      datos$Sepal.Width  < 3.35 to the left,  agree=0.813, adj=0.472, (0 split)

Node number 2: 97 observations
  predicted class=1  expected loss=0  P(node) =0.6466667
    class counts:    97     0
   probabilities: 1.000 0.000

Node number 3: 53 observations
  predicted class=2  expected loss=0  P(node) =0.3533333
    class counts:     0    53
   probabilities: 0.000 1.000
```

```
rpart.plot(arbol)
```



Analizando el dibujo del árbol, se observa que si la longitud del pétalos es menor de 3.4, nos vamos a la derecha de la rama, es decir, nos vamos al nodo verde claro que corresponde al 35% de la muestra, son flores con pétalos menores de 3.4, y el 100% pertenecen al cluster 2. Es decir el cluster 2: flores con longitud del pétalo menor de 3.4.

Si nos vamos al nodo azul entonces longitud del pétalo es mayor de 3.4, entonces, eso lo cumple, el 65% de la muestra, y el 100% de ellas son del cluster 1

6.9 Repetimos lo anterior pero con 3 clusteres

```
clusterizacion <- kmeans(datos_num,3) #Seleccionamos 2 o 3 el que escojamos como número óptir
clusterizacion
```

```
K-means clustering with 3 clusters of sizes 62, 38, 50

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     5.901613    2.748387     4.393548    1.433871
2     6.850000    3.073684     5.742105    2.071053
3     5.006000    3.428000     1.462000    0.246000
```

```
Clustering vector:
  [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [38] 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [75] 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 2 2 2 1 2 2 2 2
[112] 2 2 1 1 2 2 2 2 1 2 1 2 1 2 2 1 1 2 2 2 2 2 1 2 2 2 2 1 2 2 2 1 2 2 2 1 2
[149] 2 1

Within cluster sum of squares by cluster:
[1] 39.82097 23.87947 15.15100
 (between_SS / total_SS =  88.4 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

```
clusterizacion$cluster
```

```
  [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [38] 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [75] 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 2 2 2 1 2 2 2 2
[112] 2 2 1 1 2 2 2 2 1 2 1 2 1 2 2 1 1 2 2 2 2 2 1 2 2 2 2 1 2 2 2 1 2 2 2 1 2
[149] 2 1
```

```
table(clusterizacion$cluster)
```

```
 1  2  3
62 38 50
```

En este caso tenemos 21 flores en el cluster 1, 96 en el 2 y 33 en el 3.

```
table(datos$Species, clusterizacion$cluster)
```

```
             1  2  3
  setosa     0  0 50
  versicolor 48  2  0
  virginica  14 36  0
```

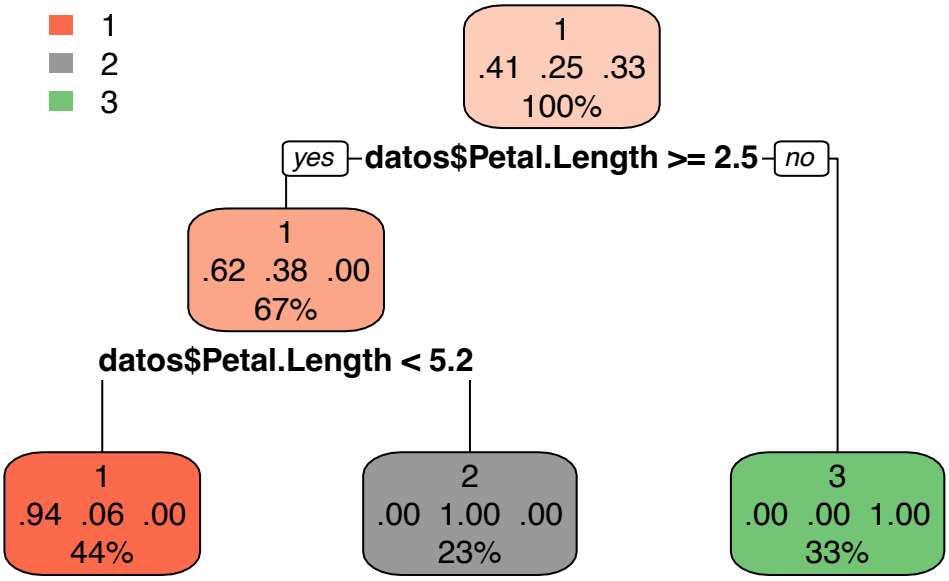
Si comparamos con la clasificación previa que tenía vemos que las flores virginica solo están en el 2, casi todas las versicolor también y las setosas se dividen entre el cluster 1 y el 3.

```
library(rpart)
library(rpart.plot)
# ahora, podemos calcular el árbol de decisión
arbol <- rpart(clusterizacion$cluster~datos$Sepal.Length+
                                      datos$Sepal.Width+
                                      datos$Petal.Length+
                                      datos$Petal.Width,
                                      method="class")
summary(arbol)
```

```
Call:
rpart(formula = clusterizacion$cluster ~ datos$Sepal.Length +
    datos$Sepal.Width + datos$Petal.Length + datos$Petal.Width,
    method = "class")
  n= 150

         CP nsplit  rel error     xerror       xstd
1 0.5681818      0 1.00000000 1.00000000 0.06853444
2 0.3863636      1 0.43181818 0.43181818 0.06053026
3 0.0100000      2 0.04545455 0.04545455 0.02242219

Variable importance
datos$Petal.Length  datos$Petal.Width datos$Sepal.Length  datos$Sepal.Width
                36                 29                 23                 13

Node number 1: 150 observations,    complexity param=0.5681818
  predicted class=1  expected loss=0.5866667  P(node) =1
    class counts:    62    38    50
   probabilities: 0.413 0.253 0.333
  left son=2 (100 obs) right son=3 (50 obs)
  Primary splits:
      datos$Petal.Length < 2.45 to the right, improve=50.96000, (0 missing)
      datos$Petal.Width  < 0.8  to the right, improve=50.96000, (0 missing)
      datos$Sepal.Length < 5.45 to the right, improve=33.82176, (0 missing)
      datos$Sepal.Width  < 3.05 to the left,  improve=22.01167, (0 missing)
  Surrogate splits:
      datos$Petal.Width  < 0.8  to the right, agree=1.000, adj=1.00, (0 split)
      datos$Sepal.Length < 5.45 to the right, agree=0.920, adj=0.76, (0 split)
      datos$Sepal.Width  < 3.35 to the left,  agree=0.833, adj=0.50, (0 split)
```

```
Node number 2: 100 observations,    complexity param=0.3863636
  predicted class=1  expected loss=0.38  P(node) =0.6666667
    class counts:    62    38     0
   probabilities: 0.620 0.380 0.000
  left son=4 (66 obs) right son=5 (34 obs)
  Primary splits:
      datos$Petal.Length < 5.15 to the left,  improve=39.604850, (0 missing)
      datos$Sepal.Length < 6.35 to the left,  improve=23.839210, (0 missing)
      datos$Petal.Width  < 1.65 to the left,  improve=22.507820, (0 missing)
      datos$Sepal.Width  < 2.95 to the left,  improve= 9.600808, (0 missing)
  Surrogate splits:
      datos$Petal.Width  < 1.95 to the left,  agree=0.85, adj=0.559, (0 split)
      datos$Sepal.Length < 6.35 to the left,  agree=0.82, adj=0.471, (0 split)
      datos$Sepal.Width  < 3.15 to the left,  agree=0.72, adj=0.176, (0 split)

Node number 3: 50 observations
  predicted class=3  expected loss=0  P(node) =0.3333333
    class counts:     0     0    50
   probabilities: 0.000 0.000 1.000

Node number 4: 66 observations
  predicted class=1  expected loss=0.06060606  P(node) =0.44
    class counts:    62     4     0
   probabilities: 0.939 0.061 0.000

Node number 5: 34 observations
  predicted class=2  expected loss=0  P(node) =0.2266667
    class counts:     0    34     0
   probabilities: 0.000 1.000 0.000
```

```
rpart.plot(arbol)
```



En este caso vemos que si la longitud del pétalo es mayor de 3.4 entonces vamos al nodo gris que mayoritariamente aparecen 2 (es decir pertenecen al clúster 2) y correspondería al 65% de la muestra. De hecho pertencen al cluster 1 con probabilidad 0.01, al 2 con probabilidad 0.99 y al 3 con probabilidad 0.

Si el pétalo es menor de 3.4 que sería un 35% del total entonces divide en función del ancho del sépalo.

- Si el ancho de sépalo es menor de 3.3 lo que corresponde a un 13% del total de la muestra entonces pertenecen al cluster 1 con probabilidad 0.95, al 2 con probabilidad 0 y al 3 con probabilidad 0.05. Mayoritariamente son flores del cluster 1.
- Si el ancho del sépalo es mayor que 3.3 lo que corresponde a un 22% entonces pertenece al cluster 1 con una probabilidad de 0.03, al 2 con una probabilidad de 0 y al 3 con una probabilidad de 0.97. Mayoritariamente son flores del cluster 3.

En resumen:

- Cluster 1 -> flores con longitud del pétalo menor de 3.4 y ancho del sépalo es menor que 3.3.
- Cluster 2 -> flores con longitud del pétalo mayor o igual que 3.4.
- Cluster 3 -> flores con longitud del pétalo menor de 3.4 y ancho del sépalo mayor o igual que 3.3.

7 Series Temporales

Una serie temporal es una secuencia de valores que se registran en momentos sucesivos, por ejemplo cada día, cada mes o cada segundo, para observar cómo evoluciona algo.

Utilizaremos el dataset de turistas en UK.

7.1 Carga de datos

```
library(readxl)
turistas_UK <- read_excel("C:\\Users\\garci\\Downloads\\turistas_UK.xlsx")
datos <- turistas_UK
View(datos)
```

7.2 Conocer los datos

```
dim(datos)
```

```
[1] 239   2
```

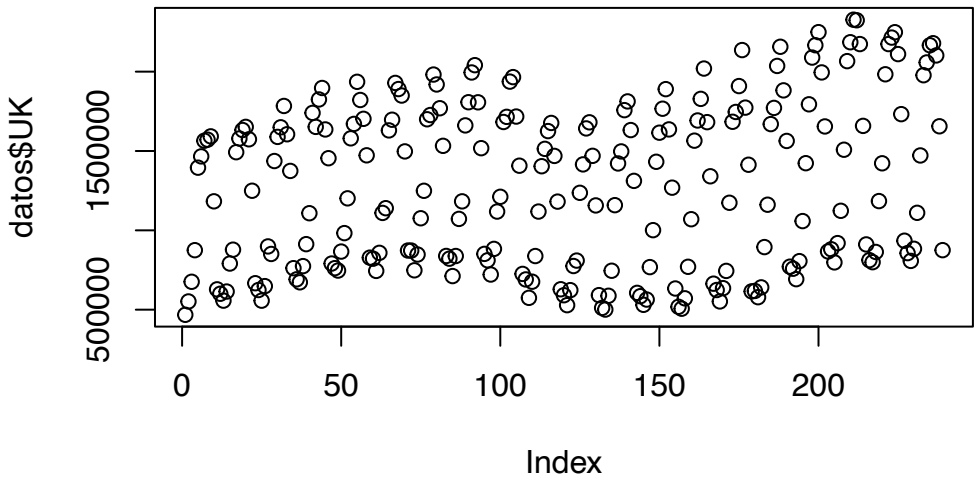
```
str(datos)
```

```
tibble [239 x 2] (S3: tbl_df/tbl/data.frame)
 $ OBS: chr [1:239] "2000M01" "2000M02" "2000M03" "2000M04" ...
 $ UK : num [1:239] 467732 551794 675759 874972 1394831 ...
```

Tenemos 239 filas y dos columnas: una columna de tipo caracter y otra de tipo número.

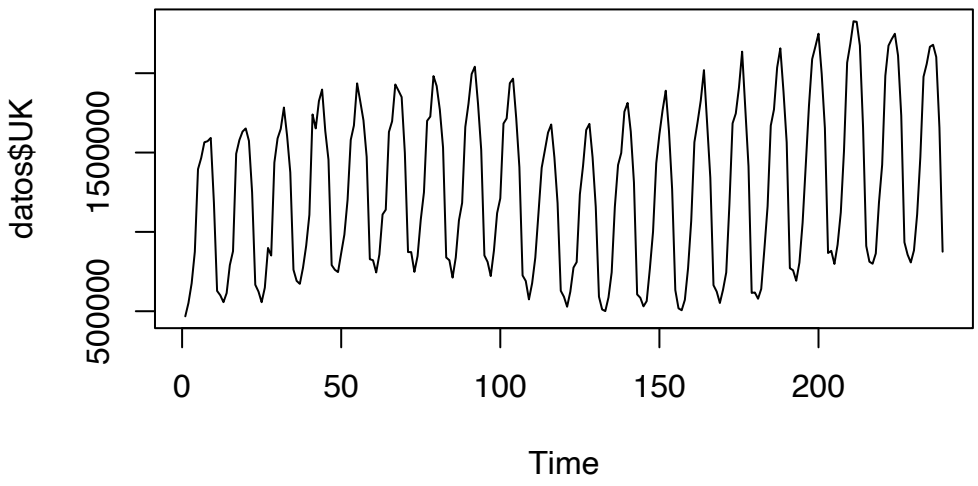
7.3 Dibujar la serie temporal

```
plot(datos$UK)
```



Al dibujarla así, sale muy raro, es debido, a que r, no sabe que UK es una variable temporal

```
plot.ts(datos$UK)
```



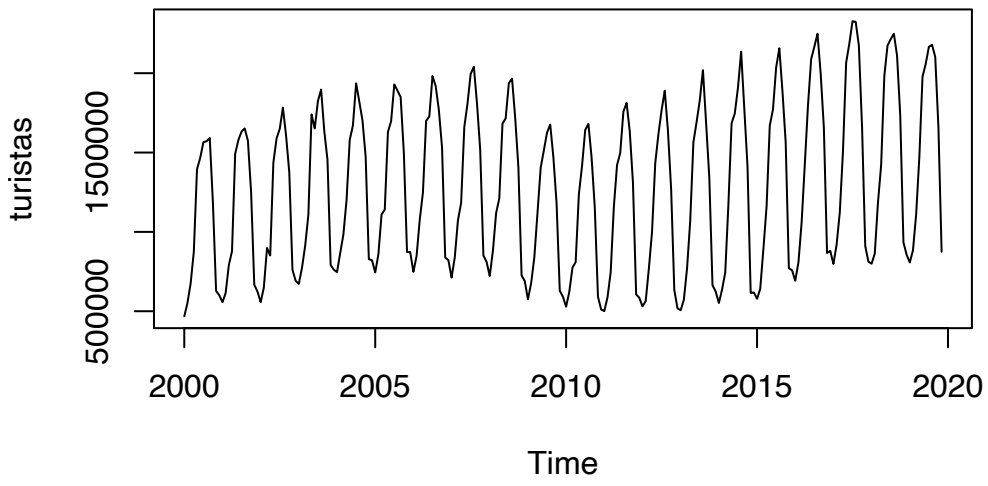
Con plot.ts ya la dibujamos como una serie temporal.

7.4 Escribir los datos como una serie temporal

Si observamos la columna OBS veremos que el primer mes que se contabiliza es Enero del año 2000 y vemos que la frecuencia es mensual así que ponemos frequency=12 de los 12 meses. Esto irá dando a cada valor de la columna UK el mes correspondiente hasta que deje de haber valores.

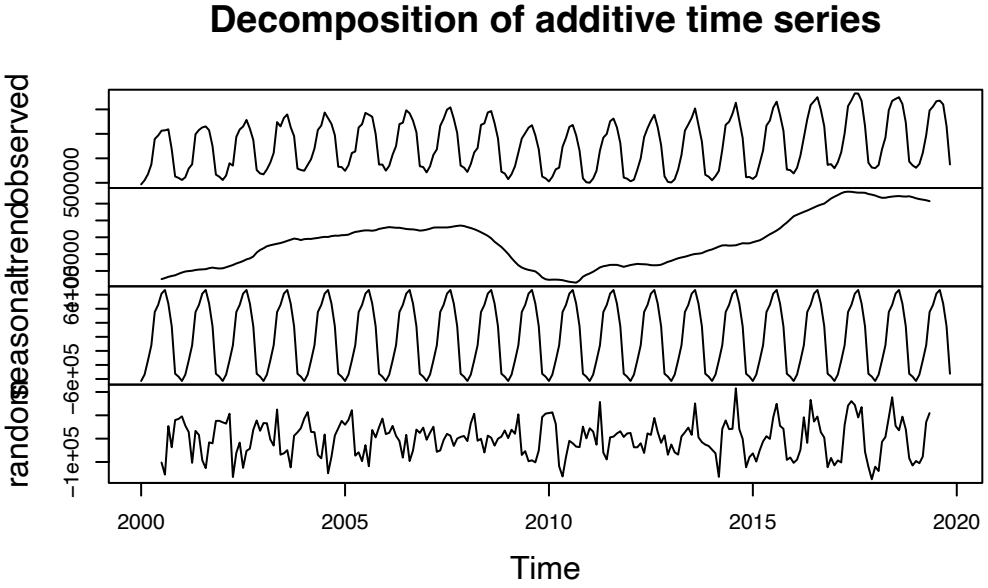
Al definirla ya como serie temporal, el propio plot la dibujará correctamente.

```
turistas <- ts(datos$UK, start = c(2000, 1), frequency = 12)
plot(turistas)
```



7.5 Extraer los componentes de una serie temporal

```
descomposicion <- decompose(turistas)
plot(descomposicion)
```



7.5.1 1. Observed (Serie original)

- Es la serie completa tal como fue registrada entre 2000 y 2020.
- Se observan **patrones repetitivos** (estacionales) y **fluctuaciones en la tendencia**.
- La amplitud varía con el tiempo, lo que sugiere que hay interacción entre componentes.

7.5.2 2. Trend (Tendencia)

- Muestra la **evolución a largo plazo**.
- Hay un **crecimiento sostenido** hasta 2007, seguido de una **caída** hasta 2011.
- Luego se recupera lentamente, lo que podría reflejar ciclos económicos, cambios estructurales o eventos externos.

7.5.3 3. Seasonal (Estacionalidad)

- Representa el **comportamiento que se repite cada año**.
- Es **estable y predecible**, lo que indica que hay patrones regulares (como ventas que suben en verano o gastos que bajan en invierno).

- Este componente es útil para ajustar modelos de predicción.

7.5.4 4. Random (Ruido o Residual)

- Contiene las **variaciones impredecibles** que no se explican por la tendencia ni la estacionalidad.
- Si el ruido es pequeño y aleatorio, el modelo de descomposición es bueno.
- Si hay patrones en el ruido, puede que falte alguna variable explicativa.

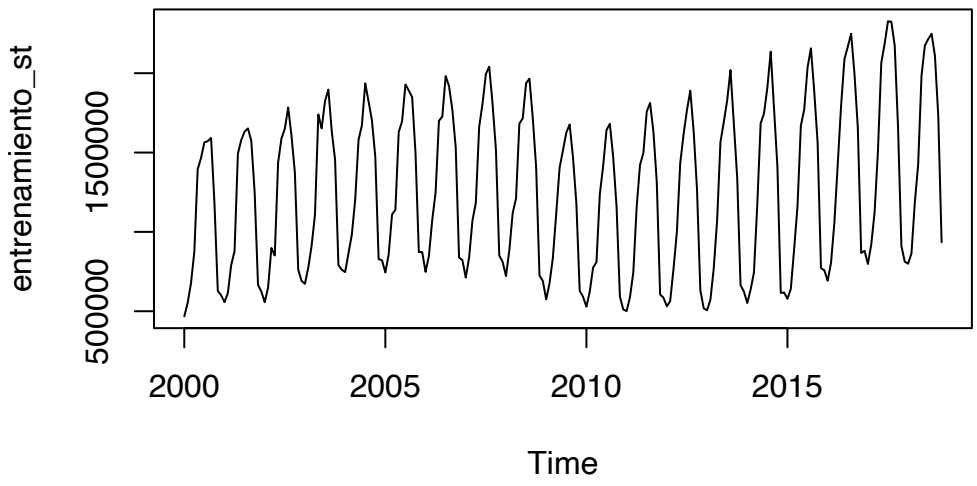
7.6 Realizar una predicción a 12 meses

7.6.1 Descomposición entre conjunto de entrenamiento y validación

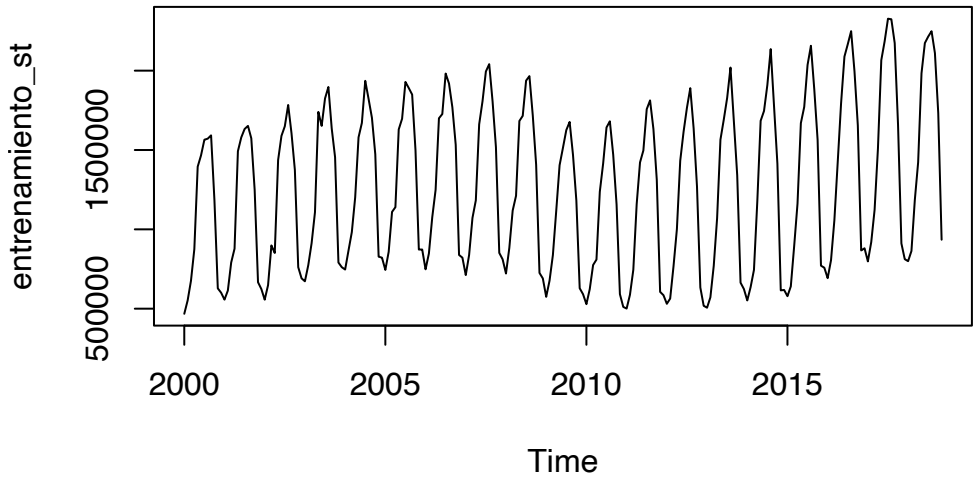
Primero si la queremos hacer correctamente dividiríamos entre conjunto de entrenamiento y conjunto de validación. Es importante el orden.

- Conjunto de validación. Serán las últimas 12 observaciones para poder luego compararlas con el resultado de la predicción por lo que será el conjunto de datos[228:239,] (filas 228 a la 239 y todas las columnas)
- Conjunto de entrenamiento. El resto de filas que se utilizan para crear la predicción (de la fila 1 a la 227) y todas las columnas [1:227,].

```
entrenamiento <- datos[1:227,]
validacion <- datos[228:239,]
entrenamiento_st <- ts(entrenamiento$UK,start = c(2000,1),frequency = 12)
plot(entrenamiento_st)
```




```
validacion_st <- ts(validacion$UK,start=c(2018,12), frequency = 12)
plot(entrenamiento_st)
```



7.6.2

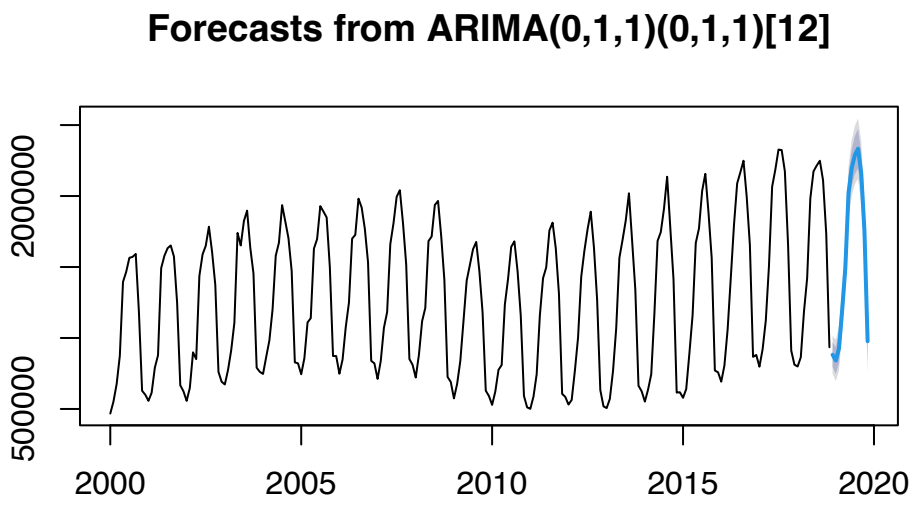
7.6.3 Aplicar la predicción

Para hacer la predicción utilizamos el modelo ARIMA, al utilizar el auto.arima no necesitamos fijar ningún parámetro, el propio ordenador lo ajusta.

```
library(forecast)
```

```
Registered S3 method overwritten by 'quantmod':
  method            from
  as.zoo.data.frame zoo
```

```
modeloarima <- auto.arima(entrenamiento_st)
prediccion12 <- forecast(modeloarima,12)
plot(prediccion12)
```

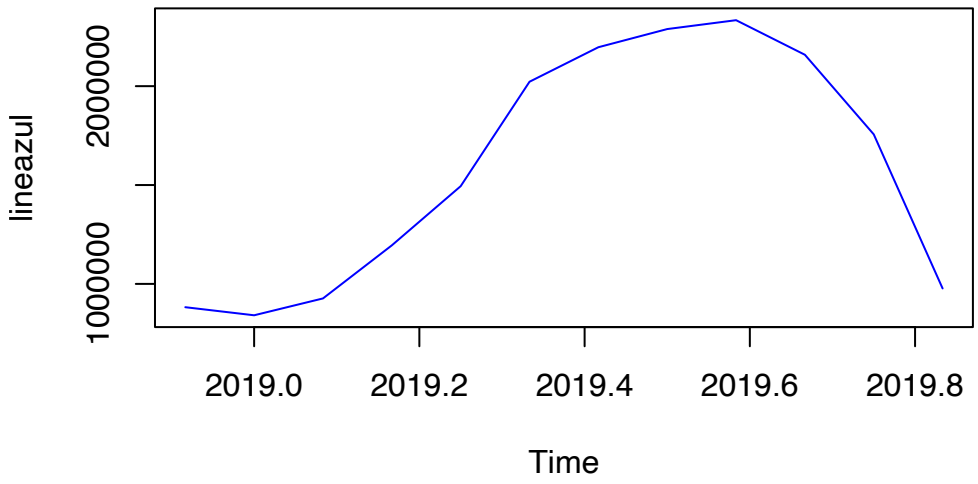


7.6.4 Evaluar la calidad de la predicción

¿la predicción es buena o es mala?

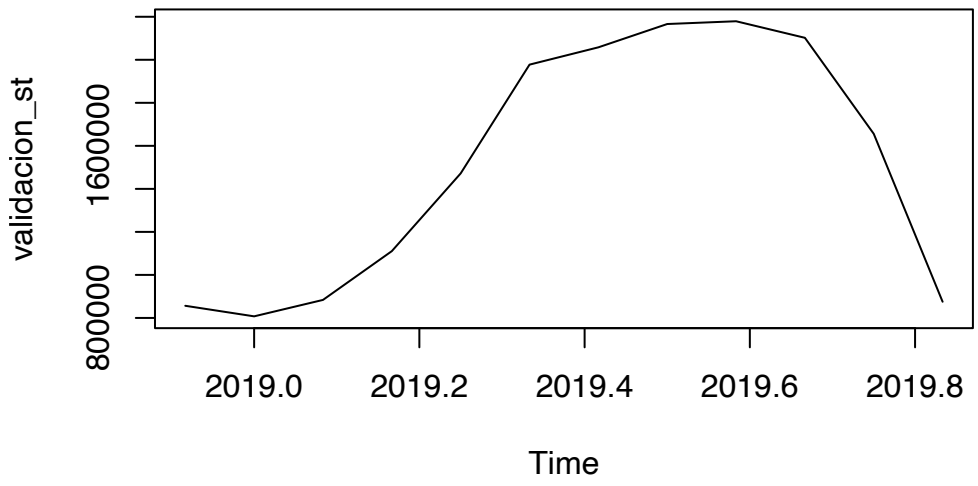
Recordemos que habíamos guardado los últimos 12 meses (validacion) por lo que podemos comparar nuestra predicción con los datos reales guardados (validacion). Nuestra predicción es la línea azul, vamos a extraerla.

```
lineazul <- prediccion12$mean
plot(lineazul, col="blue")
```



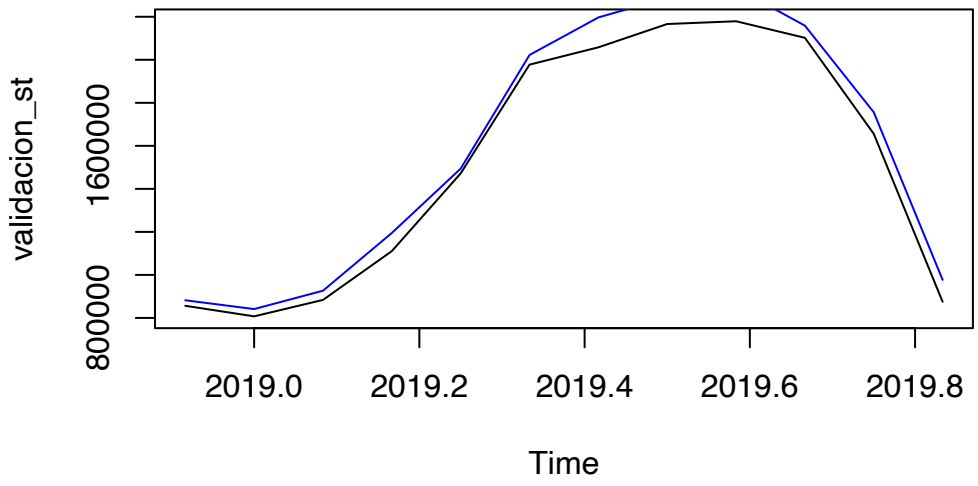
Podemos comparar nuestra predicción con los datos reales # dibujo los datos reales (que me había guardado en el cajón)

```
plot(validacion_st)
```



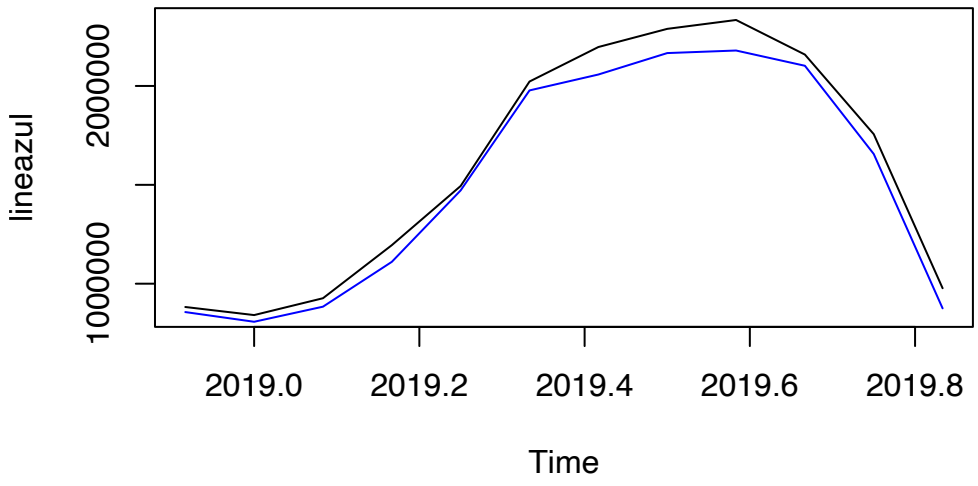
Ahora, dibujo la predicción encima lines(lineazul, col=“blue”) # se puede dibujar al reves plot(lineazul, col=“blue”) lines(validacion_st)

```
plot(validacion_st)
lines(lineazul, col="blue")
```



Se puede hacer también al revés

```
plot(lineazul)
lines(validacion_st , col="blue")
```



Gráficamente, parece que la predicción es bastante buena (pero es a "ojo"), podemos calcular las métricas utilizando la librería forecast, donde el comando accuracy nos calcula las métricas.

```
accuracy(validacion_st,lineazul)
```

```
                ME     RMSE      MAE      MPE     MAPE      ACF1 Theil's U
Test set 77356.07 89000.03 77356.07 4.946191 4.946191 0.3292721 0.2458223
```

Por supuesto,el error de la predicción, es la diferencia entre la linea azul y la negra. Cuanto menores sean las métricas, mejor es la predicción (el error es menor)

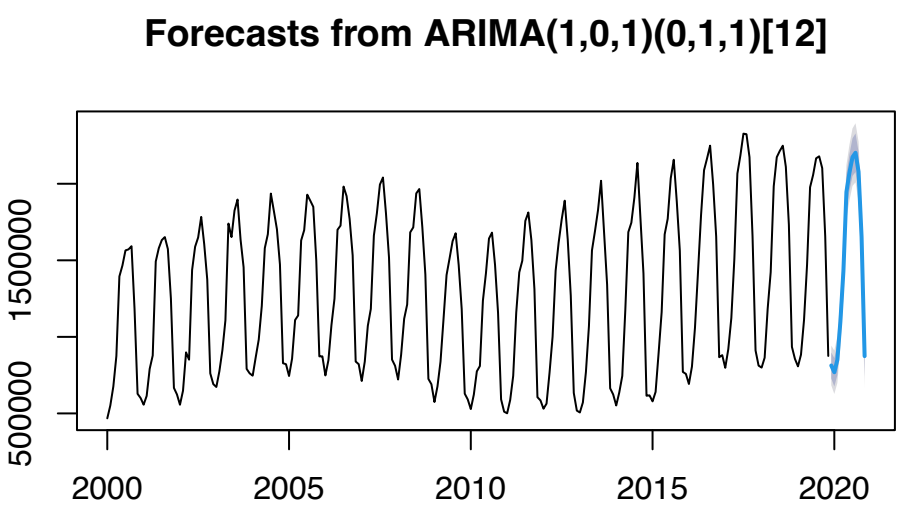
```
accuracy(lineazul,validacion_st)
```

```
                 ME     RMSE      MAE       MPE     MAPE      ACF1 Theil's U
Test set -77356.07 89000.03 77356.07 -5.272334 5.272334 0.3292721 0.2470711
```

7.7 Predicción general (sin dividir en entrenamiento y validación)

Si el modelo es bueno, ahora me puedo permitir hacer una predicción para los próximos 12 meses con una cierta "seguridad"

```
datos_st <- ts(datos$UK, start=c(2000,1), frequency = 12)
modelo_total <- auto.arima(datos_st)
prediccion_total12 <- forecast(modelo_total,12)
plot(prediccion_total12)
```


Forecasts from ARIMA(1,0,1)(0,1,1)[12]
500000
1500000
2000
2005
2010
2015
2020