# EJERCICIOS

### **Sección 2. Introducción a R**

1.-Crea un vector denominado ventas con los valores 120, 140, 90

2.-Asigna estos valores “enero”, “febrero”, “marzo”, como nombres de las columnas

3.-Calcula la media de las ventas de los 3 meses

4.-Crea un nuevo vector con los elementos de ventas mayores de 100

5.-Calcular cual es el mes de mayor número de ventas

**ventas<-c(120,140,90)**

**meses<-c(‘enero’,’febrero’,’marzo’)**

**names(ventas)<-meses**

**mean(ventas)**

**filtro<- ventas > 100**

**ventas.mayores.100 <- ventas[filtro]**

**ventas.mayores.100**

**filtro2<- ventas == max(ventas)**

**filtro2**

**ventas[filtro2]**

### **Sección 3. Matrices en R**

1.-Crear un matriz con 2 filas y 4 columnas, con estos dos vectores, vector1=(4,5,3) vector2=(2,8,25,14)

2.-Crear una matriz con 6 filas y 4 columnas con números del 1 al 24, utilizando la expresión 1:24 y nrow

3.-Seleccionar el valor de la fila 3 y la columna 4 de la matriz

4.-Crear una nueva matriz seleccionando las 3 primeras filas y las 2 primeras columnas

5.-A esta nueva matriz, añadirle una columna que sea la suma de los valores de cada fila

**vector1 <- c(4,5,3)**

**vector2 <- c(2,8,25,14)**

**matriz <- rbind(vetor1,vector2)**

**matriz**

**matriz2<-matrix(1:24, nrow=6)**

**matriz2**

**matriz[3,4]**

**matriz3<-matriz[1:3,1:2]**

**matriz3**

**suma=rowSums(matriz.nueva)**

**suma**

**matriz<-cbind(matriz.nueva, suma)**

**matriz**

### **Sección 4. Data Frames en R**

## Ejercicio 1.

**Crear un Data Frame con los siguientes datos de 3 personas con su nombre, edad y sexo**

**Antonio, 40, H**

**Maria, 30, M**

**Juan, 35, M**

**nombre<-c(‘Antonio’,’Maria’,’Juan’)**

**edad<-c(40,30,35)**

**sexo<-c(‘H’,’M’,’H’)**

**datos<-data.frame(nombre,edad,sexo)**

**datos**

## Ejercicio 2.

a) Crear la variable coches que almacene los datos del dataset mtcars

**coches<-mtcars**

b) Verificar si coches es un dataframe

**is.data.frame(coches)**

**o**

**str(coches)**

c) Imprimir las 6 lineas de la variable coches

**head(coches)**

**head(coches,10) #imprime las 10 primeras líneas**

**head(coches, n=10L) #imprime las 10 primeras líneas**

d) Cual es la media de la columna mpg

**mean(datos$mpg)**

e) Cual es el valor de cyl para el coche ‘Mazda RX4’

**coches[‘Mazda RX4’, ]$cyl**

f) Selecciona las filas de coches donde la columna cyl es igual a 8 y la columna disp. > 400

**coches[coches$cyl==8 & coches$disp>400, ]**

**subset(coches, cyl==8 & disp>400)**

g) Selecciona las columnas disp y wt

**coches[,c(‘disp’,’wt’)]**

h) Crear una nueva columna, rendimiento que sea igual a hp/wt

**coches$rendimiento<-coches$hp/coches$wt**

### **Sección 5. Listas en R**

## Ejercicio 1.

**vector<-1:30**

**matriz <-matrix(1:30, nrow=3)**

**datos<-mtcars**

**lista<-list(vector,matriz,datos)**

**lista[[1]] # devuelve los datos almacenados en vector**

**lista[[2]] # devuelve los datos almacenados en matriz**

**lista[[3]] # devuelve los datos almacenados en datos**

**lista<-list(vector=vector, matriz=matriz,datos=datos)**

**lista**

**$vector #devuelve los datos almacenados en vector**

**$matriz**

**$datos**

**lista[‘vector’]**

**$vector #devuelve vector**

**class(lista$vector) #devuelve su tipo, en este caso “integer”**

**str(lista) # devuelve información detallada sobre nuestra lista**

### **Sección 6. Entrada y salida de datos en R mediante RStudio**

## Ejercicio 1. Archivos CSV

**matriz<-matrix(1:16, nrow=4)**

**matriz**

**write.csv(matriz, file=’matriz.csv’)**

**datos<-read.csv(‘matriz.csv’)**

**datos$X<-NULL #nos cargamos la fila que añade correlativa**

**class(datos) #devuelve data.frame**

**head(datos)**

**datos$suma<-datos$V1+datos$V2+datos$V3+datos$v4 #añado una nueva columna como suma de las cuatro que tiene el data frame**

**write.csv(datos, file=’datos.csv’)**

## Ejercicio 2. Archivos XLSX

**install.package(‘readxl’) #sólo la primera vez**

**library(readxl)**

**excel\_sheets(‘poblacion.xlsx’)**

**datos <- excel\_sheets(‘poblacion.xlsx’, sheet=’Hoja 1’)**

**datos$continente #continente = nombre de la columna**

**str(datos) #devuelve un tribble [nxm]**

**excel <-lapply (Excel\_sheets(‘poblacion.xlsx’),read\_excel, path=’poblacion.xlsx’)**

**excel**

**datos<-excel[[1]] #devuelve los datos de la hoja 1**

**install.packages(‘xlsx’) #maneja ficheros Excel para guardarlos**

**library(xlsx)**

**datos<-head(mtcars)**

**datos**

**class(datos)**

**write.xlsx(datos,’fichero2.xlsx’)**

**datos$continente**

**str(datos) #devuelve tribble [nxm]**

## Ejercicio 3. Archivos DBF

**install.packages(‘RSQLite’)**

**library(RSQLite)**

**con<-dbConnect(RSQlite::SQLite(),”basededatos.db”)**

**head(mtcars)**

**dbWriteTable(con,”coches”,mtcars) #conexion, nombre de la tabla, datos que metemos**

**dbReadTable(con,”coches”)**

**res<-dbSendQuery(con,”SELECT \* FROM coches WHERE cyl=8”) #creo Query**

**dbfFetch(res) # ejecuto la Query**

**dbClearResult(res) # libero la Query (vacío el resultado obtenido)**

**dbDisconnect(con) # libero la conexión con la base de datos**

### **Sección 7. Programación básica en R**

## 1. Operadores lógicos en R

**< menor**

**> mayor**

**& y**

**== igual**

**| o**

**! no**

**datos<-mtcars**

**head(datos)**

**subset(datos, mpg>20 & cyl==6)**

**subset(datos, !(cyl==6)) o subset(datos, !=cyl=6)**

## 2. Sentencias condicionales if else

**#usamos RScripts en el Notebook de R**

**numero<-20**

**if (numero > 20) {**

**print (‘el numero es mayor que 10’)**

**} else {**

**Print (‘el numero es menor que 10’)**

**}**

**numero<-8**

**if (numero > 10){**

**print (‘el numero es mayor que 10’)**

**} else if (numero > 5){**

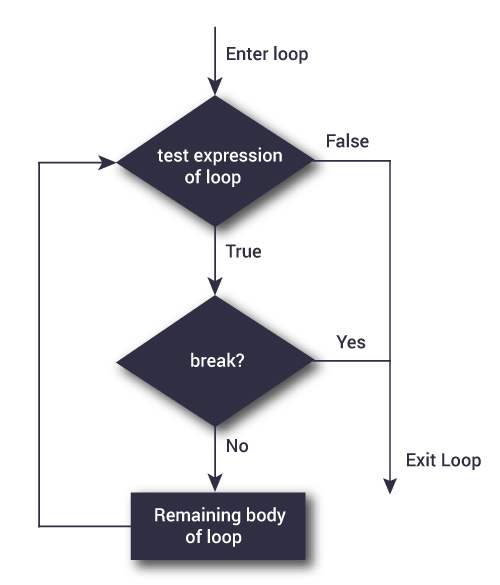
**print (‘el número es mayor que 5 y menor que 10’)**

**} else {**

**print (‘el numero es menor que 5’)**

**}**

## 3. Bucle while

****

**numero<-1**

**while (numero < 10){**

**texto <- paste(‘El numero es ’,numero)**

**print (texto)**

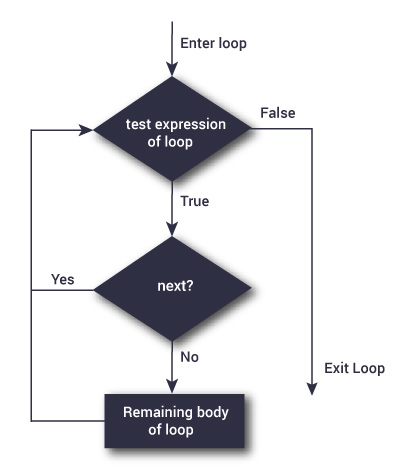
**if (numero == 5) {**

**break**

**}**

**numero <- numero +1**

**}**

****

**x <- 1:5**

**for (val in x) {**

**if (val == 3){**

**next**

**}**

**print(val)**

**}**

## 4. Bucle for

**lista<-list(12,c(1,2,3),mtcars) #lista de 3 variables 12, dataframe 1 a 3 y mtcars**

**lista**

**for (elemento in lista){**

**print (elemento)**

**}**

**matriz<-matrix(1:9, nrow=3)**

**for (elemento in matriz){**

**print(elemento)**

**}**

**vector<-c(1,2,3,4)**

**for (elemento in vector){**

**print(elemento)**

**}**

## 5. Funciones en R

**hola <- function(){**

**print (‘Hola como estás’)**

**}**

**hola() #La llamamos por su nombre**

**## 2ª versión---------------------------------------------**

**hola <- function(nombre){**

**texto<-paste(‘hola como estas ‘,nombre)**

**print (texto)**

**}**

**hola(‘Juan’) #La llamamos por su nombre**

**[1] “hola como estas Juan”**

**## 3ª versión ---------------------------------------------**

**sumar <-function(numero1,numero2){**

**resultado<-numero1+numero2**

**return(resultado)**

**}**

**print(sumar(3,5))**

**[1] 8**

## Ejercicio 1

**# determinar si un número es o no par**

**par <- function(valor) {**

**if (valor/2 == valor%%2){. #alternativa valor%%2==0**

**print(paste('El numero ',valor,' es par'))**

**} else {**

**print (paste('El numero ',valor,' es impar'))**

**}**

**}**

## Ejercicio 2

**# verificar si una variable es numerica o carácter**

**verifica <-function(valor){**

**if (is.numeric(valor)){**

**print(paste(valor,' es una variable de tipo numérico'))**

**}else if (is.character(valor)){**

**(print(paste(valor, ' es una variable de tipo carácter')))**

**}else {(print(paste(valor,' no es ni variable ni carácter')))**

**}**

**}**

## Ejercicio 3

**# valor máximo de los valores de un vector**

**vector <-c(1,3,4,17,23,9,98)**

**vmax<-0**

**for (numero in vector){**

**if(numero>vmax){**

**vmax<-numero**

**}**

**}**

**print(vmax)**

## Ejercicio 4

**# media de dos valores**

**media2numeros <-function (numero1, numero2){**

**media <-(numero1+numero2)/2**

**return(media)**

**}**

## Ejercicio 5

**#matriz cuadrada**

**n <- 3**

**m<- n\*n**

**matriz <-matrix(1:m, ncol=n, byrow=TRUE)**

**suma<-0**

**for(filas in 1:n){**

**for (columnas in 1:n){**

**suma<-suma+matriz[filas,columnas]**

**}**

**}**

**print(suma)**

### **Sección 8. Programación avanzada en R**

## 1. Funciones predefinidas

**seq(1:10)**

**[1] 1 2 3 4 5 6 7 8 9 10**

**seq(1:10, by=2)**

**[1] 1 3 5 7 9**

**vector<-c(1,8,4,5,2,7,3,6)**

**sort(vector) #ordena el vector**

**[1] 1 2 3 4 5 6 7 8**

**vector2<-c(‘b’,’d’,’c’,’a’)**

**sort(vector2)**

**[1] a b c d**

**summary( ) # proporciona información sobre la estructura de datos**

**summary(mtcars)**

**rev(vector2) #invierte el vector2**

**[1] d c b a**

**str(vector2)**

**devuelve la estructura del vector2**

**vector1<-c(1,2,3,4)**

**vector2<-c(5,6,7)**

**vector3<-append(vector1,vector2)**

**vector3**

**[1] 1 2 3 4 5 6 7**

**is.numeric(vector3)**

**[1] FALSE**

**vector<-c(1:12)**

**is.vector(vector)**

**[1] TRUE**

**Is.list(vector)**

**[1] FALSE**

**lista<-as.list(vector)**

## 2. Aplicar funciones sobre un vector o una lista

**vector<-c(1,2,3)**

**suma<-function(numero{**

**resultado<-numero+10**

**return(resultado)**

**}**

**print(vector)**

**vector2<-sapply(vector,suma) #aplica sobre el vector la función suma**

**print(vector2)**

**lista<-list(1,2,3)**

**aleatorio<-function(numero){**

**numero.aleatorio<-sample(1:100,1) #1 numero aleatorio entre 1-100**

**resultado<-numero+numero.aleatorio**

**return(resultado)**

**}**

**print(lista)**

**lista2<-lapply(lista,aleatorio) #aplica la función aleatorio a lista**

sapply == sobre vectores

lapply == sobre listas

## 3. Funciones anónimas en R

**vector<-c(1,2,3,4)**

**vector2 <- sapply (vector, function(numero){numero+2}**

**vector2**

**[1] 3 4 5 6 # la función añade 2 a cada valor**

## 4. Funciones matemáticas en R

**abs(-2)**

**[1] 2**

**vector<-c(1,-3,4,-5,5)**

**abs(vector)**

**[1] 1 3 4 5 5**

**sum(1,2,3)**

**[1] 6**

**matriz<-matrix(1:12, nrow=3)**

**sum(matriz)**

**[1] 78**

**mean(matriz)**

**[1] 6.5**

**M**

**mean(vector)**

**[1] 3.6**

**valores= sample(1:100,12)**

**[1] 8 48 9 93 58 45 97 96 21 69 99 52 #12 numeros aleatorios entre 1 y 100**

**matriz2 <- matrix(valores, nrow=3)**

**min(matriz2)**

**[1] 8**

**round(3.14159264389,4)**

**[1] 3.1415**

## 5. Expresiones regulares

**texto<-‘Hola que tal’**

**grepl(‘que’,texto)**

**[1] TRUE #la cadena que aparece**

**grep(‘que’,texto)**

**[1] 1 # apaparece en la línea 1**

**vector <- C(1,7,2,3,4,5)**

**grep(3,vector)**

**[1] 4 #está en la posición 4 del vector**

**grepl(3,vector)**

**[1] FALSE FALSE FALSE TRUE FALSE FALSE #recorre el vector**

## 6. Manipular la fecha y la hora en R

**fecha<-Sys.data()**

**fecha**

**[1] “2020-05-13”**

**fecha2<- ‘2020-01-10’**

**class(fecha2)**

**[1] “carácter”**

**fecha2<-as.Date(fecha2)**

**fecha3<-as.Date(“04-01-2020”, format=”%d-%m-%Y”)**

**help(“strptime”)**

**Sys.time() #fecha y hora**

**format(Sys.time(), format”%H:%M:%S”)**

**strptime(“12:00:00”, format”%H:%M:%S”)**

**[1] “2020-05-13 12:00:00 CEST”**

### **Sección 9. Manipulación de datos con R**

## 1. Manipulación de datos con dplyr

**install.packages(“dplyr”)**

**install.packages(“mycflights13”)**

**library(dplyr)**

**library(mycflights13)**

**filter(flights, month==4, day==10) #filtra el por mes 4 y dia 10 del dataset**

**selects(flights, origin, dest, air\_time) #select flights por los 3 campos del dataset**

**slices(flights, 1:5) #muestra las filas 1 a 5 del dataset flights**

**arrange(flights, year, month, day, df\_time) #ordena el dataset por los campos dados**

**arrange(fligths, year, month, day, df\_time, desc(dep\_time) #ordena descendentemente por el campo dep\_time**

**rename(flights, aerolínea=carrier) #cambia el nombre en vista a carrier x aerolínea**

**distint(select(flights,carrier)) #muestra los valores de carrier de forma única**

**sumarise(flights, media=mean(air\_time), na.rm=TRUE) # calcula la media obviando los valores nulos**

**Sample\_n(flights, 10) # muestra 10 filas aleatorias del datase flights**

## 2. Operador pipe

**Pipe %>%**

**coches<-mtcars**

**resultado<-arrange(sample\_n(filter(coches,mpg>20),size=5),desc(mpg))**

**resultado**

**# también puede realizarse en pasos:**

**paso1<-filtro(coches,mpg>0)**

**paso2<-sample\_n(paso1)**

**paso3<-arrange(paso2, desc(mpg))**

**resultado<-paso3**

**resultado**

**# o con pipes**

**resultado<-coches%>%filtro(mpg>20)%>%sample\_n(size=5)%>%arrange(desc(mpg))**

## 3. Limpieza de datos con tidyr

**install.packages(“tidyr”)**

**install.packages(“data.table”)**

**library(tidyr)**

**library(data.table)**

**gether #agrupa varias columnas en 1 sóla**

**año<-c(2018,2019,2020)**

**enero<-c(120,100,80)**

**febrero<-c(80,90,140)**

**marzo<-c(110,150,70)**

**datos<-data.frame(año=año,enero=enero,febrero=febrero,marzo=marzo)**

**datos**

**datos<-gether(datos,Mes,Ingresos, enero:marzo)**

**datos**

**[1]. año Mes Ingresos**

**…….**

**#spread es la función contraria a gether, separa 1 columna en varias**

**datos<-spread(datos,Mes,Ingresos)**

**datos2<-daaframe(columna=c(‘a.b’,’c.d’,’e.f’)**

**datos2<-separate(datos2, columna, c(c1,c2))**

**datos2<-unite(datos2, columna, c1,c2,sep=’.’)**

**unite == separa**

### **Sección 10. Visualización de datos en R con RStudio**

## 1.-Histogramas

**install.packages(‘ggplot2’)**

**library(ggplot2)**

**install.packages(‘ggplot2movies’)**

**library(ggplot2movies)**

**películas<-movies**

**head(películas)**

**películas[c(‘title’,’year’,’rating’)]**

**datos<-ggplot(películas,aes(x=rating))**

**histograma <-datos+geom\_histogram()**

**print(histograma) #muestra el histograma**

**histograma<-datos+geom\_histogram(binwidth=0.3)**

**print(histograma) #incluye más clases**

**histograma<-datos+geom\_histogram(binwidth=0.3, color=’greeen’)**

**print(histograma)**

**histograma<- datos+geom\_histogram(binwidth=0.3, color=’greeen’, fill=green)**

**print(histograma)**

**histograma<- datos+geom\_histogram(binwidth=0.3, color=’greeen’, fill=Green, alpha=0.3) #transparencia**

**print(histograma)**

**histograma<- datos+xlab(‘Puntuación’)+ylab(‘Frecuencia’)+ggtitle(‘Histograma)) #titulos de histograma y ejes**

**print(histograma)**

## 2.-Scatterplots

**Dos variables**

**library(ggplot2)**

**coches<-mtcars**

**head(coches)**

**grafico<-ggplot(coches, aes(x=disp., y=mpg))**

**grafico<-grafico+geom\_pint()**

**print(grafico)**

**graficos<-grafico+geom\_point(size=8, alpha=0.4)**

**print(grafico)**

**grafico<-grafico+geom\_point(size=8, color=’red’) #color https://www.color-hex.com**

**print(grafico)**

**grafico<-grafico+geom\_point(size=8, color=’#2717C4’) #color https://www.color-hex.com**

**print(grafico)**

**grafico<-grafico+geom\_point(aes(size=wt))**

**print(grafico)**

**grafico<-ggplot(coches, aes(x=disp,y=mpg))**

**grafico<-grafico+geom\_point(size=8,des(color=hp))**

**grafico<-grafico+scales\_color(gradient(low=’blue’,high=’red’)**

**print(grafico)**

## 3.-Barplots

**library(ggplot2)**

**datos<-mpg**

**head(datos)**

**str(datos)**

**grafico<-ggplot(datos,aes(x=class))**

**grafico<-grafico+geom\_bar()**

**print(grafico)**

**grafico<-grafico+geom\_bar(color=’red’,fill=’blue’)**

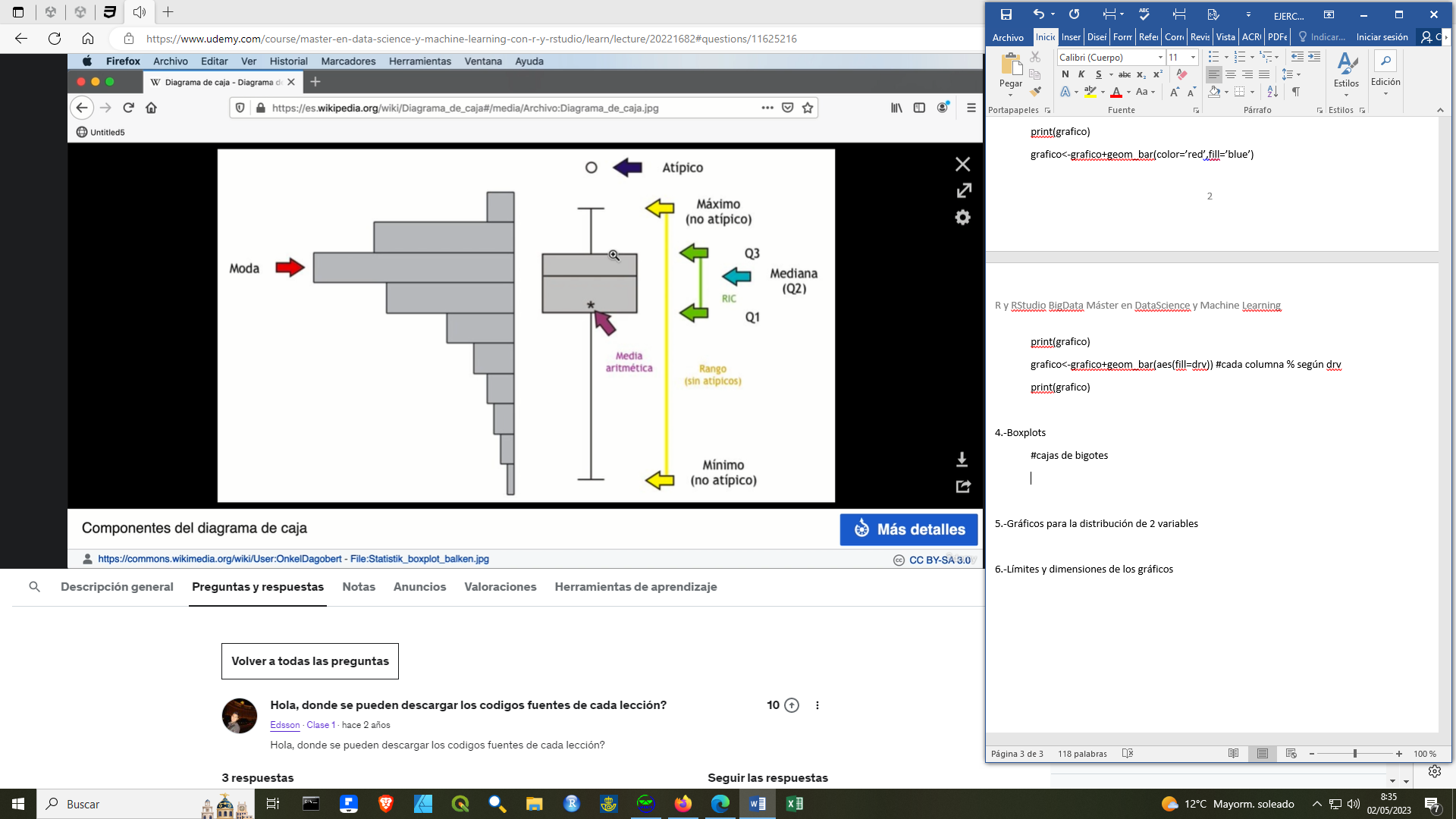
**print(grafico)**

**grafico<-grafico+geom\_bar(aes(fill=drv)) #cada columna % según drv**

**print(grafico)**

## 4.-Boxplots

**#cajas de bigotes**

****

**library(ggplot2)**

**datos<-mtcars**

**head(datos)**

**grafico<-ggplot(datos, aes(x=factor(cyl),y=mpg,))**

**grafico<-grafico+geom\_boxplot()**

**print(grafico)**

**grafico<-grafico+geom\_boxplot(fill=’green’)**

**print(grafico)**

**grafico<-grafico+geom\_boxplot(aes(fill=Factory(cyl)))**

**print(grafico) #muestra cada factor(cyl) en un color**

**grafico<-grafico+theme\_dark() #existen varios theme\_ el light es el por defecto**

**print(grafico)**

## 5.-Gráficos para la distribución de 2 variables

**library(ggplot2)**

**library(ggplo2movies)**

**películas<-movies**

**head(películas)**

**grafico<-ggplot(películas,aes(x=year,y=rating))**

**grafico<-grafico+geom\_bind2d()**

**print(grafico)**

**grafico<-grafico+scale\_fill\_gradient(low=’blue’,high=’red’) #grafico de densidad**

**print(grafico)**

**grafico<-ggplot(películas,aes(x=year,y=rating)) #curvas de nivel**

**grafico<-grafico+geom\_density2d()**

**print(grafico)**

## 6.-Límites y dimensiones de los gráficos

**library(ggplot2)**

**datos<-mpg**

**head(datos)**

**str(datos)**

**grafico<-ggplot(datos,aes(x=displ, y=hwy))**

**grafico<-grafico+geom\_point()**

**print(grafico)**

**grafico<-grafico+coord.\_cartesian(xlim=c(2,5),ylim=c(20,30)) #delimito el intervalo**

**print(grafico) #subset del gráfico previo**

**grafico<-ggplot(datos,aes(x=displ, y=hwy))**

**grafico<-grafico+geom\_point()**

**print(grafico)**

**grafico<-grafico+cood\_fixed(ratio=1/3)**

**print(grafico)**

### **Sección 11. Ejercicios de visualización de datos**

## 1.-Ejercicio 1

Reproducir un histograma del dataset mpg, ejex = cty y en el ejey=count

## 2.-Solución al Ejercicio 1

library(ggplot2)

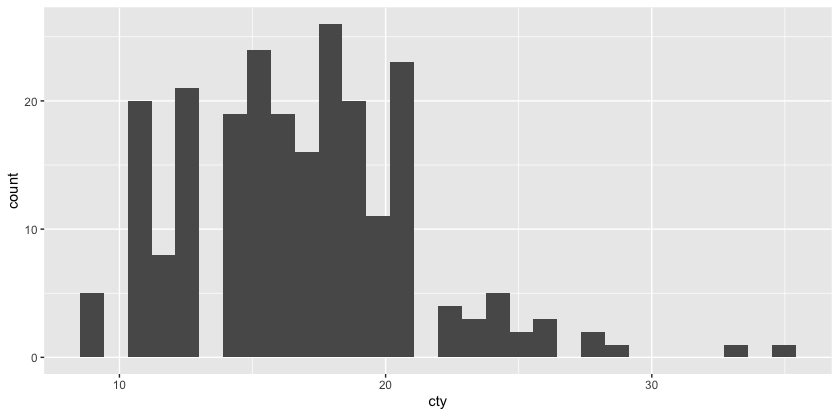
datos<-mpg

head(datos)

grafico<-ggplot(datos,aes(x=cty))

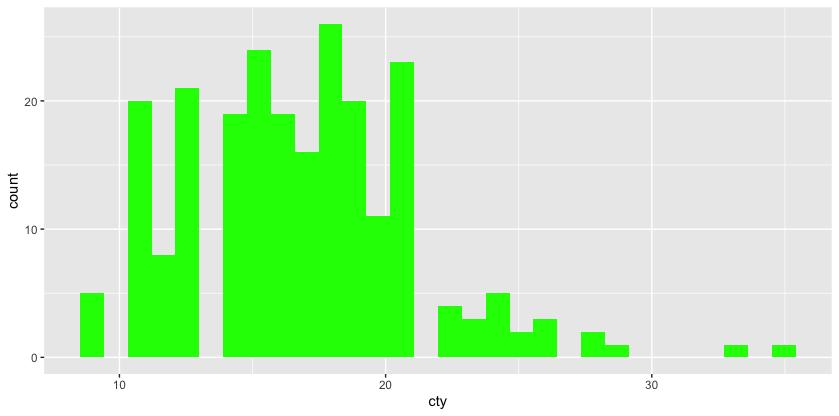
grafico<-grafico+geom\_histogram()

print(grafico)



grafico<-grafico+geom\_histogram(fill=”green”, alpha=0.5, bins=30)

print(grafico)



grafico<-grafico+ggtile(‘Dataset = mpg’)

print(grafico)

## 3.-Ejercicio 2

A partir del dataset mpg crear un histograma de barras acumulada en función del factor (cyl)

## 4.-Solución al Ejercicio 2

library(ggplot2)

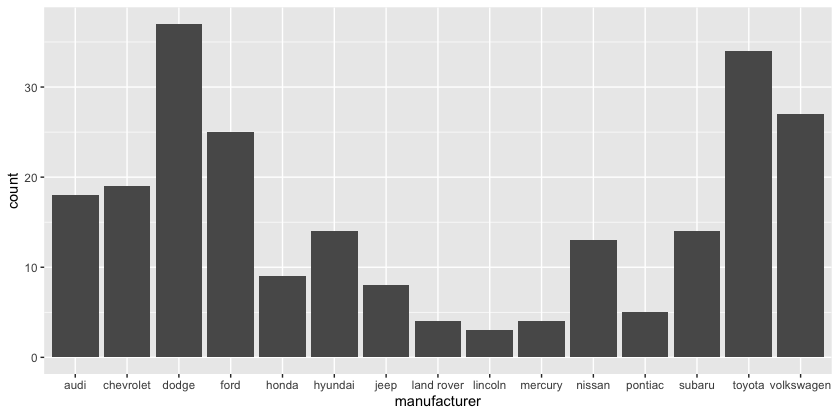
datos<-mpg #1 variable y tipo discreta

head(datos)

grafico<-ggplot(datos, aes(x=manufacturer))

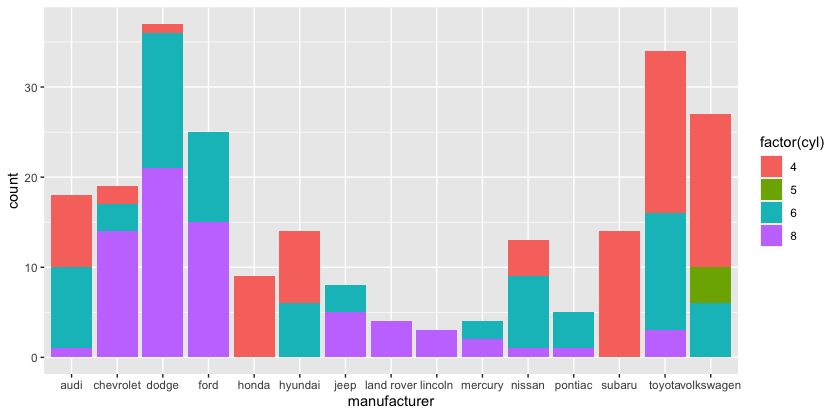
grafico<-grafico+geom\_bar()

print(grafico)



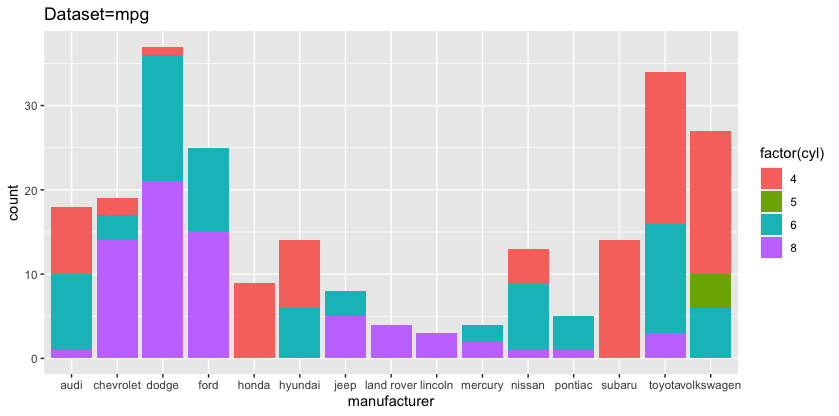
grafico<-grafico+geom\_bar(aes(fill=factor(cyl)))

print(grafico)



grafico<-grafico+ggtitle(‘Dataset=mpg’)

print(grafico)



## 5.-Ejercicio 3

Dado el dataset txhousing construir un gráfico de dispersión de sales vs. volumes

## 6.-Solución al Ejercicio 3

#2 variables continiuas que

library(ggplot2)

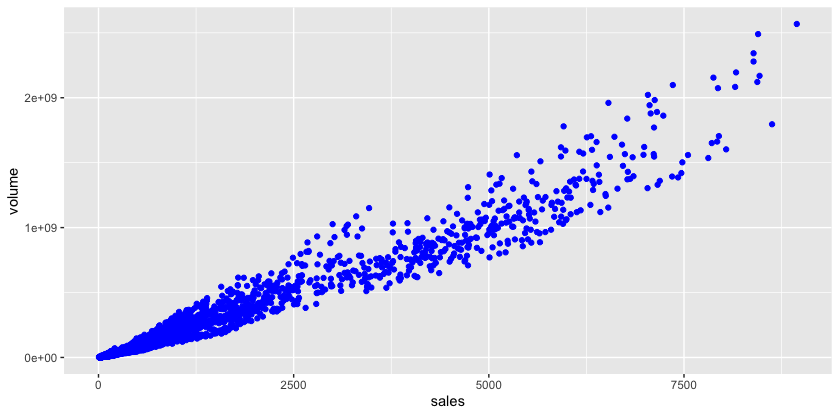
datos<-txhousing

head(datos)

grafico<-ggplot(datos, aes(x=sales, y=volume))

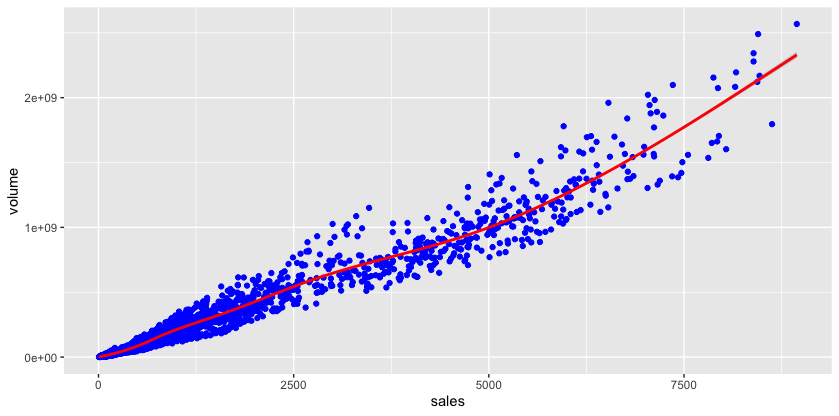
grafico<-grafico+geom\_point(color=’blue’, alpha=0.4)

print(grafico)



grafico<-grafico+geom\_smooth(color=’red’)

print(grafico)



# EJERCICIOS

### **Sección 12. Gráficos interactivos con Plotly**

### 1.-Plotly

install.packages(‘plotly’)

library(plotly)

library(ggplot2)

datos<-mtcars

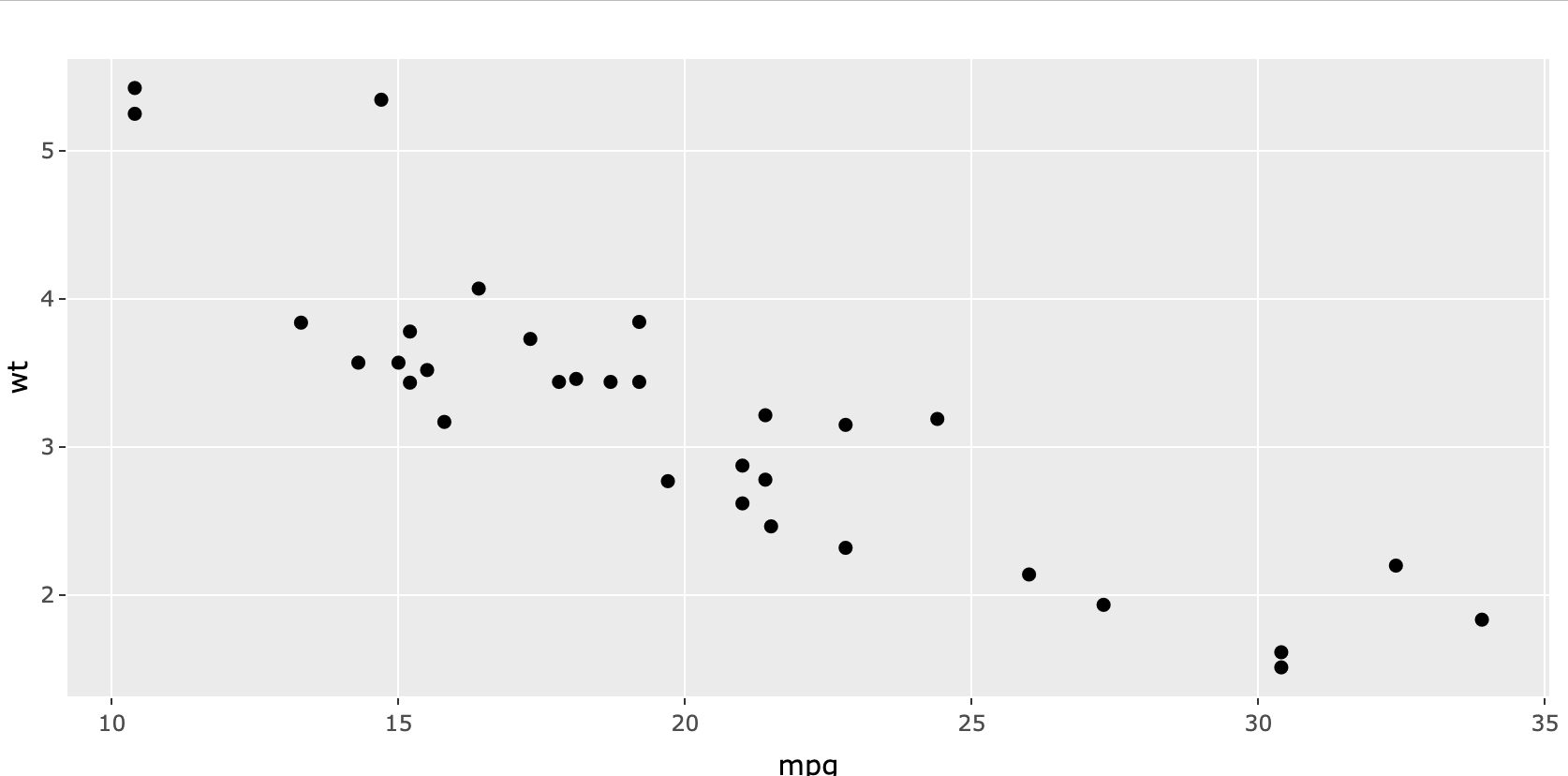
head(datos)

grafico🡨ggplot(datos,aes(mpg,wt))+geom\_point()

print(grafico)

grafico2<-ggplotly(grafico)

print(grafico2)



**Default bubble chart** #https://plotly.com/ggplot2

library(plotly)

library(dplyr)

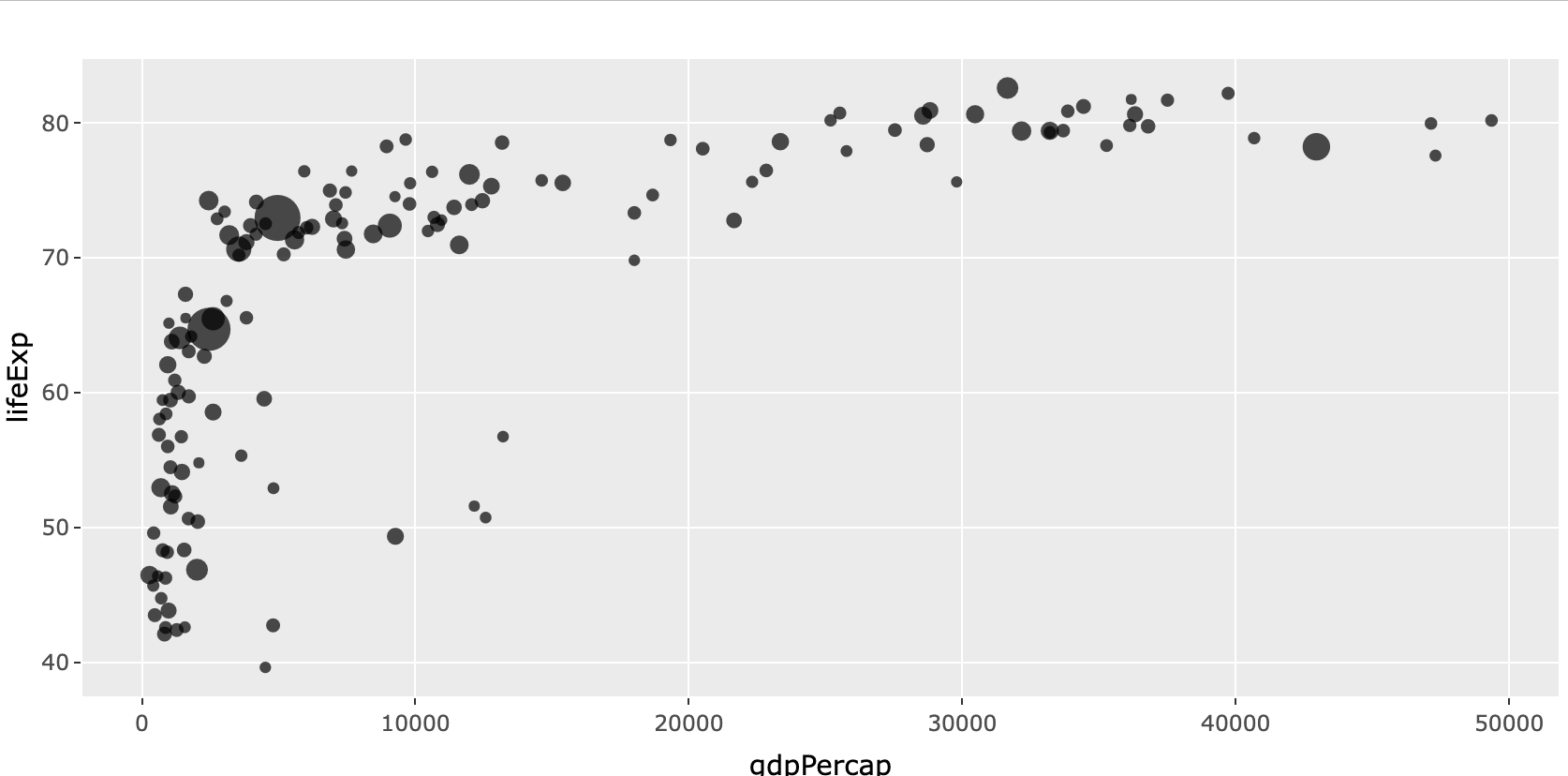
library(gapminder)

data <- gapminder %>% filter(year=="2007") %>% dplyr::select(-year)

p <- ggplot(data, aes(x=gdpPercap, y=lifeExp, size = pop)) +

geom\_point(alpha=0.7)

ggplotly(p)



### **Sección 13. Introducción al Machine Learning**

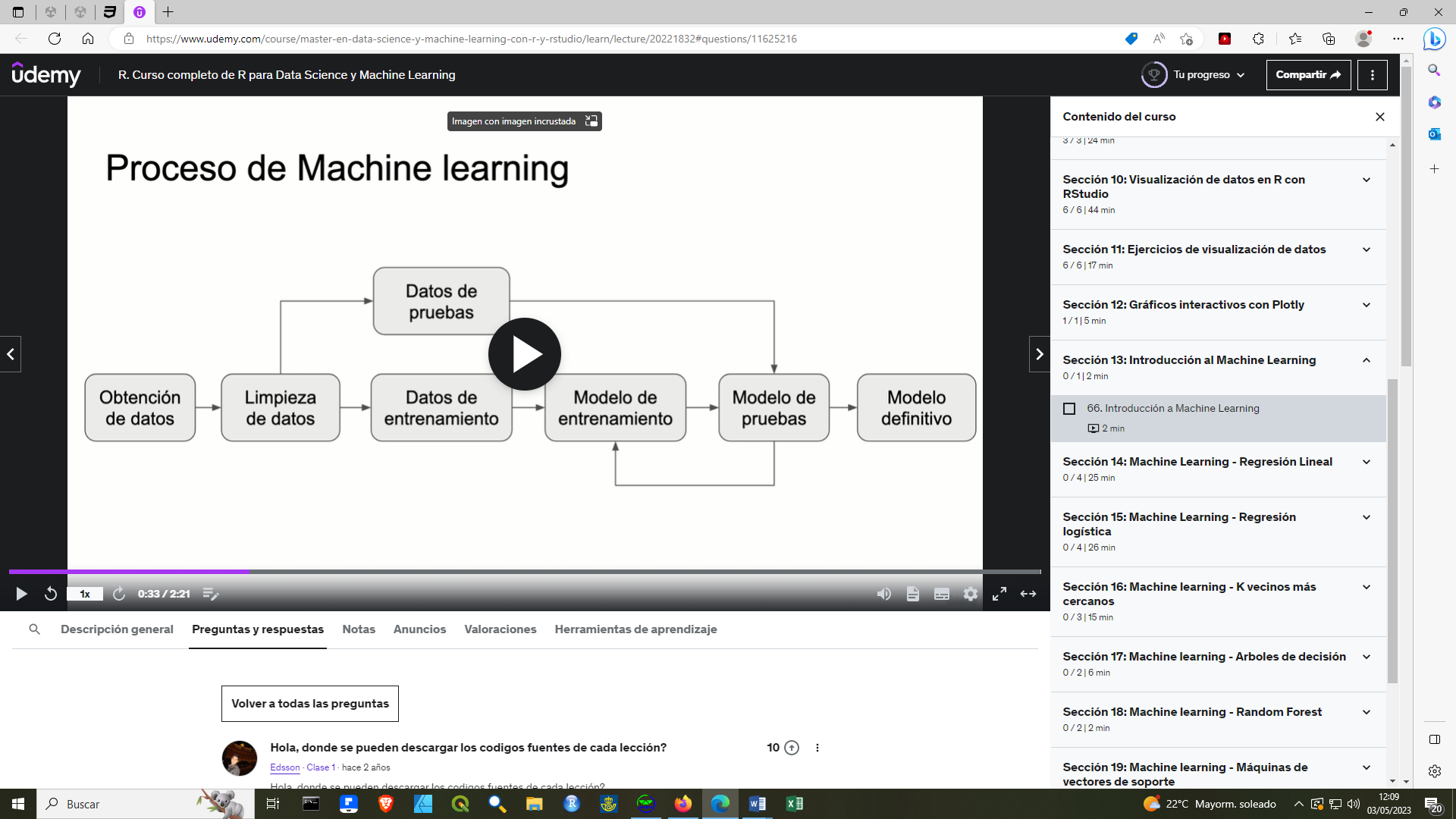
### 1.-Introduccción a Machine Learning

# Machine Learning

## Aprendizaje de máquinas

### Aprendizaje automático

* Machine learning o aprendizaje de máquinas o aprendizaje automático, es el subcampo de las ciencias de la computación y una rama de la inteligencia artificial, cuyo objetivo es desarrollar técnicas que permitan que las computadoras aprendan.
* Machine learning en un método de análisis de datos que automatiza la construcción de un modelo analítico.
* Machine learning permite a los ordenadores encontrar soluciones a problemas, sin ser explícitamente programados para ello, gracias al uso de algoritmos, que aprenden de los datos.



## Tipos de algoritmos de Machine Learning

Aprendizaje supervisado: este algoritmo necesita datos previamente etiquetados (solucionados) para aprender a realizar el trabajo. En base a estos datos, el algoritmo es capaz de aprender a resolver problemas futuros similares.

Aprendizaje no supervisado: este algoritmo necesita indicaciones previas, que le enseñan a comprender y analizar la información, para resolver problemas futuros similares. No necesita datos previamente etiquetados.

Aprendizaje de refuerzo: este algoritmo aprende por su cuenta, en base a unos conocimientos previamente introducidos y a la práctica que realiza sobre los problemas, aprendiendo en función del éxito o fracaso que obtiene al resolver los problemas.

### **Sección 14. Machine Learning. Regresión Lineal**

### 1.-Introduccción a la regresión lineal en Machine Learning

### Regresión lineal

La regresión lineal es un algoritmo de aprendizaje supervisado que se utiliza en Machine Learning y en estadística.

La regresión líneal es una aproximación para modelar la relación entre una variable escalar dependiente “y” y una o más variables explicativas “x”.

La idea es dibujar una recta que indicará la tendencia del conjunto de datos.



Repositorio de machine learning: <https://archive.ics.uci.edu/ml/datasets.php>

### 2.-Ejemplo de regresión lineal – parte 1

student-mat.csv #dataframe que vamos a emplear

datos<-read.csv('student-mat.csv',sep=';')

head(datos)

any(is.na(datos))

**[1] FALSE**

library(ggplot2)

ligrary(ggthemes)

library(dplyr)

columnas.numericas<-sapply(datos, is.numeric)

head(columnas.numericas)

school sex age address famsize Pstatus

FALSE FALSE TRUE FALSE FALSE FALSE

datos.correlacion<-cor(datos[,columnas.numericas])

print(datos.correlacion)

age Medu Fedu traveltime studytime failures famrel freetime goout Dalc Walc

age 1.000000000 -0.163658419 -0.163438069 0.070640721 -0.004140037 0.24366538 0.053940096 0.01643439 0.126963880 0.131124605 0.11727605

Medu -0.163658419 1.000000000 0.623455112 -0.171639305 0.064944137 -0.23667996 -0.003914458 0.03089087 0.064094438 0.019834099 -0.04712346

Fedu -0.163438069 0.623455112 1.000000000 -0.158194054 -0.009174639 -0.25040844 -0.001369727 -0.01284553 0.043104668 0.002386429 -0.01263102

………….

#devuelve matriz de correlaciones de las variables numéricas

install.packages(‘corrgram’)

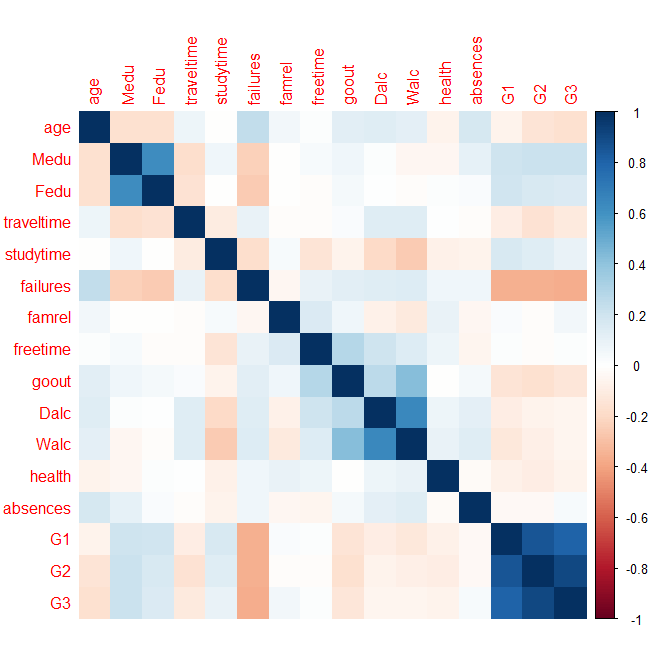
install.packages(‘corrplot’)

library(corrplot)

library(corrgram)

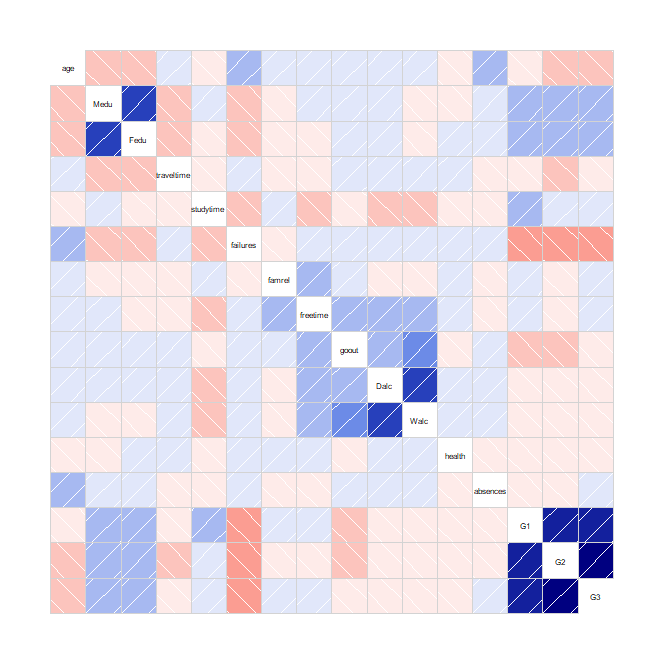
grafico<-corrplot(datos.correlacion,method='color')

#matriz de correlaciones por colores



corrgram (datos)

#matriz de correlaciones visual



ggplot(datos, aes(x=G3))+geom\_histogram(bins=20,alpha=0.5,fill=’blue’)

#histograma para la variable G3

### 3.-Ejemplo de regresión lineal – parte 2

install.packages(‘caTools’)

library(caTools)

set.seed(80)

ejemplo<-sample.split(datos$G3,SplitRatio=0.7)

entrenamiento<-subset(datos, ejemplo==TRUE)

#entrenamos al modelo

pruebas<-subset(datos,ejemplo==FALSE)

#observaciones para el modelo

modelo<-lm(G3~. , entrenamiento)

#variable correlaciona G3 con todas las demás, ella incluida “.”

print(summary(modelo))

#muestra un sumario del modelo

#debemos observar el p-valor si <0.05 hipotesis nula

#R-squared cuanto más próxima a 1 mejor se ajusta el modelo

residuos<-residuals(modelo)

class(residuos)

residuos<-as.data.frame(residuos) #convertimos la variable en un dataframe

head(residuos) #los visualizamos

ggplot(residuos,aes(residuos))+geom\_histogram(fill=’blue’,alpha=0.5)

### 4.-Ejemplo de regresión lineal – parte 3

plot(modelo)

<Enter> para ver el próximo gráfico:

<Enter> para ver el próximo gráfico:

<Enter> para ver el próximo gráfico:

<Enter> para ver el próximo gráfico:

predicciones<-predict(modelo,pruebas)

resultados<-cbind(predicciones,pruebas$G3) #creo una matriz

head(resultados)

colnames(resultados)<-c(‘prediccion’,’real’)

resultados<-as.data.frame(resultados)

head(resultados)

min(resultados)

**[1] -1.423384**

cero<-function(x){

if (x<0){

return(0)

} else{

return(x)

}

}

resultados$prediccion<-apply(resltados$prediccion,cero)

min(resultados$prediccion

**[1] 0**

error<-mean(resultados$real-resultados$prediccion)^2)

error

**[1] 4.547965**

sse<-sum((resultados$prediccion-resultados$real)^2)

sst<-sum((mean(datos$G3)-resultados$real)^2)

resultado<-1-sse/sst

resultado

[1] 0.777195

### **Sección 15. Machine Learning. Regresión Logística**

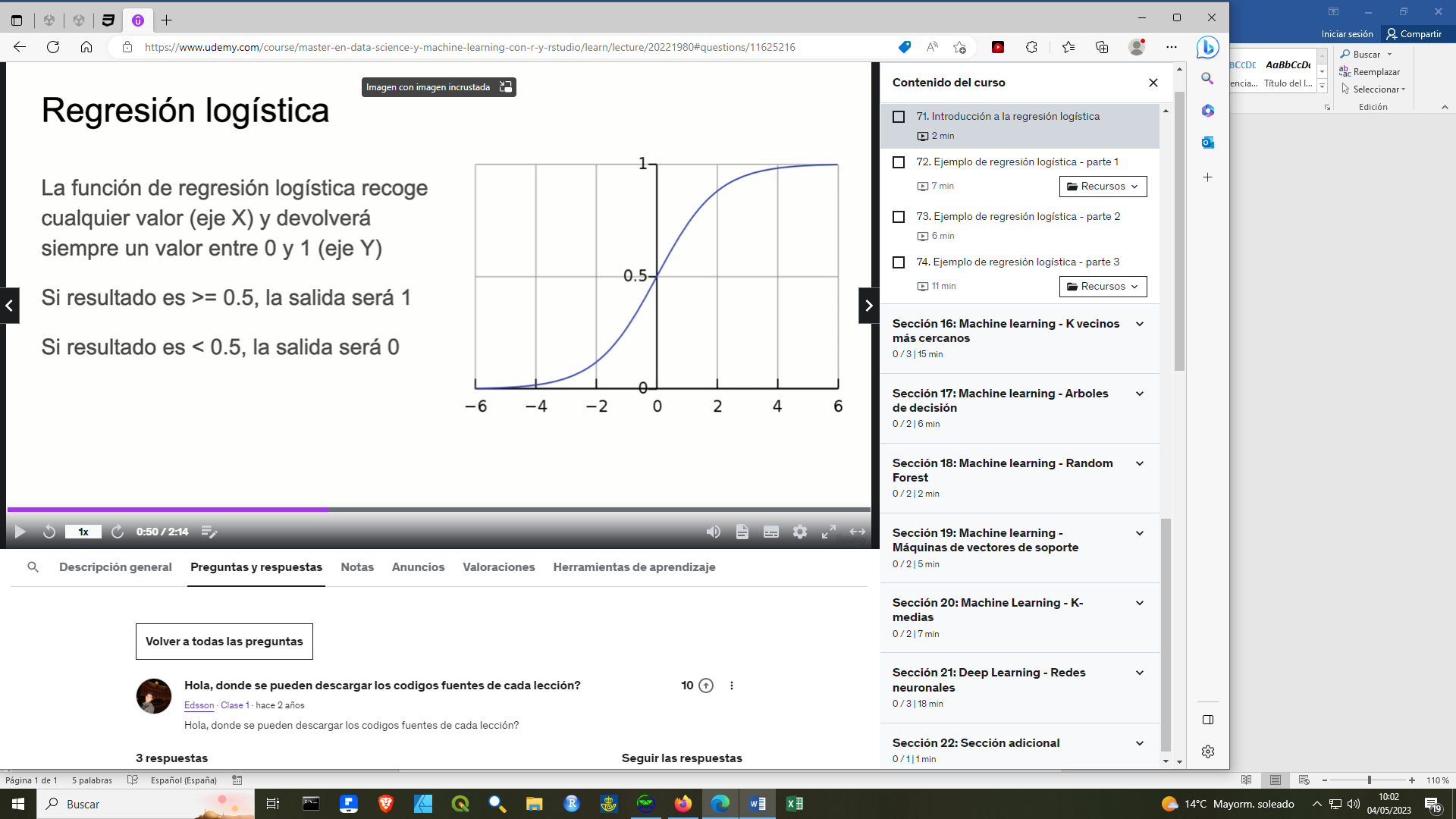
### 1.-Introduccción a la regresión logística

La regresión logística es un tipo de análisis de regresión, utilizado para predecir el resultado de una variable categórica (una variable que puede adoptar un número limitado de categorías) en función de otras variables independientes.

Es útil para modelar la probabilidad de que un evento pueda ocurrir en función de otros factores.

Es un método de clasificación, por ejemplo, para clasificar los correos según sean válidos o no, para clasificar a las personas que solicitan un préstamo según lo puedan pagar o no, o para clasificar a las personas según tengan una enfermedad concreta.

Estos ejemplos, son ejemplos de clasificaciones binarias, en las que sólo hay dos categorías (si o no) (categoría 1 o categoría 2)



La función de regresión logística recoge cualquier valor (eje x) y devolverá siempre un valor entre 0 y 1 (eje y)

Si el resultado es >=0.5, la salida será 1.

Si el resultado es <0.5, la salida será 0.

Una matriz de confusión sirve para evaluar nuestro modelo de regresión logística.

Ejemplo de una matriz de confusión para unas pruebas de detección de spam.

PC (positivos correctos), NC (negativos correctos)

FP (falsos positivos, **error tipo 1**), FN (falsos negativos, **error tipo 2**)

|  |  |  |
| --- | --- | --- |
| Correos totales =200 | Predicción (SI es SPAM) | Predicción (NO es SPAM) |
| Valor real = Si es SPAM | 30 (PC) | 15 (FN) |
| Valor real = NO es SPAM | 5 (FP) | 150 (NC) |

La precisión sirve para saber la probabilidad de acierto en la predicción:

**Precisión** = (Positivos correctos + Negativos correctos) /Total

**Precisión** = (30+150) /200=0.9 (probabilidad de acierto del 90%)

La tasa de error sirve para saber la probabilidad de error en la predicción

**Tasa de error** = (Falsos positivos + Falsos negativos) /Total

**Tasa de error** = (5+15) /200 =0.1 (probabilidad de error del 10%)

### 2.-Ejemplo de regresión logística – parte 1

datos<-read.csv(‘titanic.csv’)

head(datos)

PassengerId Survived Pclass Name Sex Age SibSp Parch

1 1 0 3 Braund, Mr. Owen Harris male 22 1 0

2 2 1 1 Cumings, Mrs. John Bradley (Florence Briggs Thayer) female 38 1 0

3 3 1 3 Heikkinen, Miss. Laina female 26 0 0

4 4 1 1 Futrelle, Mrs. Jacques Heath (Lily May Peel) female 35 1 0

5 5 0 3 Allen, Mr. William Henry male 35 0 0

6 6 0 3 Moran, Mr. James male NA 0 0

Ticket Fare Cabin Embarked

1 A/5 21171 7.2500 S

2 PC 17599 71.2833 C85 C

3 STON/O2. 3101282 7.9250 S

4 113803 53.1000 C123 S

5 373450 8.0500 S

6 330877 8.4583 Q

str(datos)

'data.frame': 891 obs. of 12 variables:

$ PassengerId: int 1 2 3 4 5 6 7 8 9 10 ...

$ Survived : int 0 1 1 1 0 0 0 0 1 1 ...

$ Pclass : int 3 1 3 1 3 3 1 3 3 2 ...

$ Name : chr "Braund, Mr. Owen Harris" "Cumings, Mrs. John Bradley (Florence Briggs Thayer)" "Heikkinen, Miss. Laina" "Futrelle, Mrs. Jacques Heath (Lily May Peel)" ...

$ Sex : chr "male" "female" "female" "female" ...

$ Age : num 22 38 26 35 35 NA 54 2 27 14 ...

$ SibSp : int 1 1 0 1 0 0 0 3 0 1 ...

$ Parch : int 0 0 0 0 0 0 0 1 2 0 ...

$ Ticket : chr "A/5 21171" "PC 17599" "STON/O2. 3101282" "113803" ...

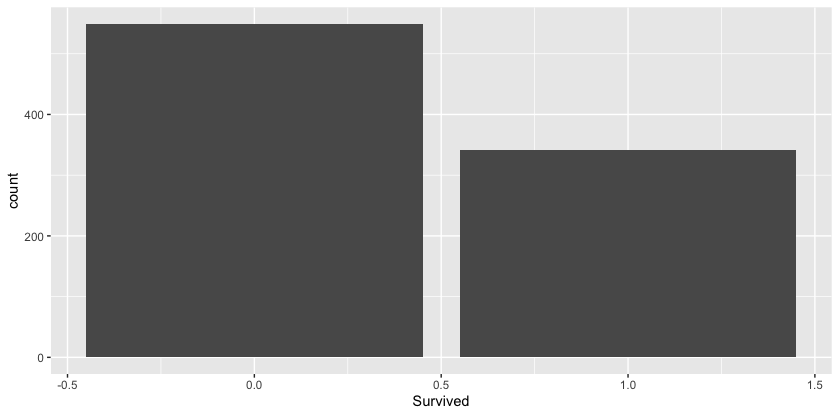
$ Fare : num 7.25 71.28 7.92 53.1 8.05 ...

$ Cabin : chr "" "C85" "" "C123" ...

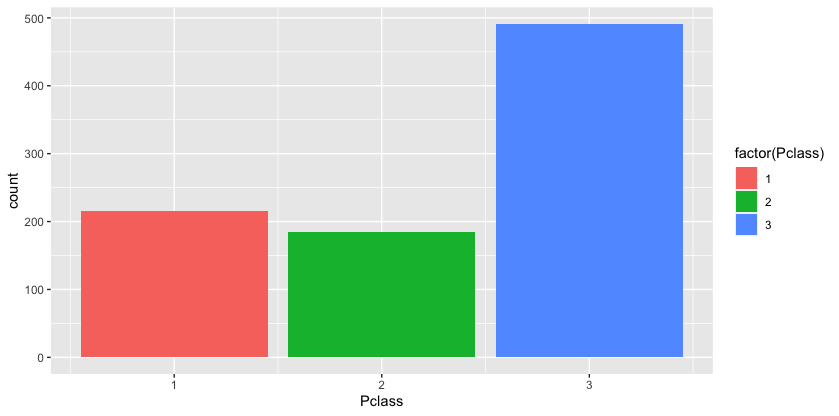
$ Embarked : chr "S" "C" "S" "S" ...

library(ggplot2)

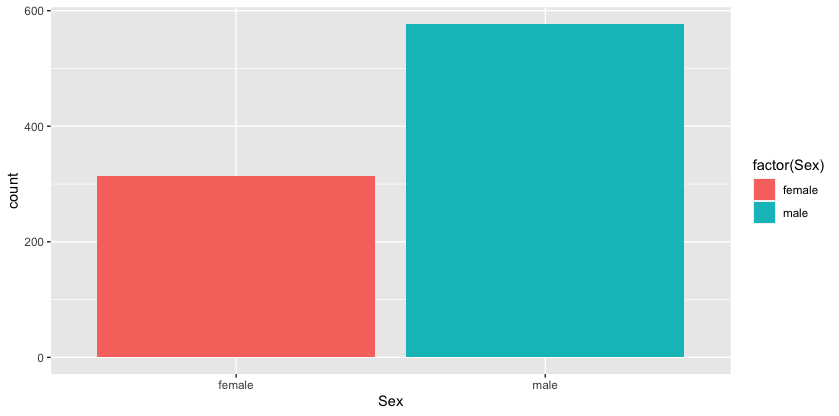
ggplot(datos,aes(Survived))+geom\_bar() #histograma de vivos/fallecidos



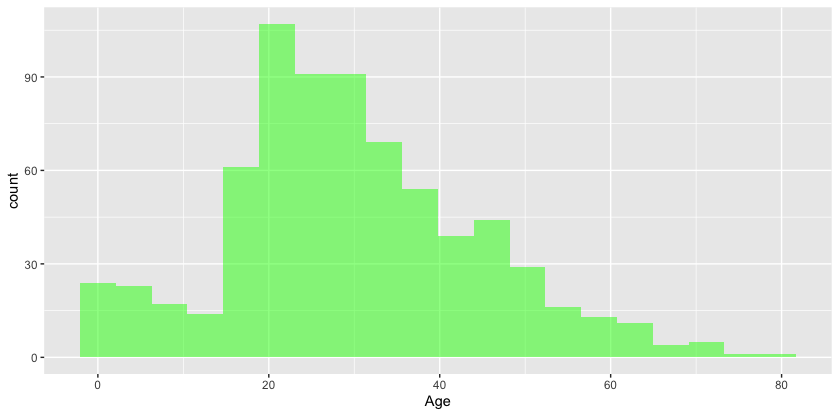
ggplot(datos,aes(Pclass))+geom\_bar(aes(fill=factor(Pclass))) #histograma por pasajeros vs. clase



ggplot(datos, aes(Sex))+geom\_bar(aes(fill=factor(Sex))) #histograma por sexo



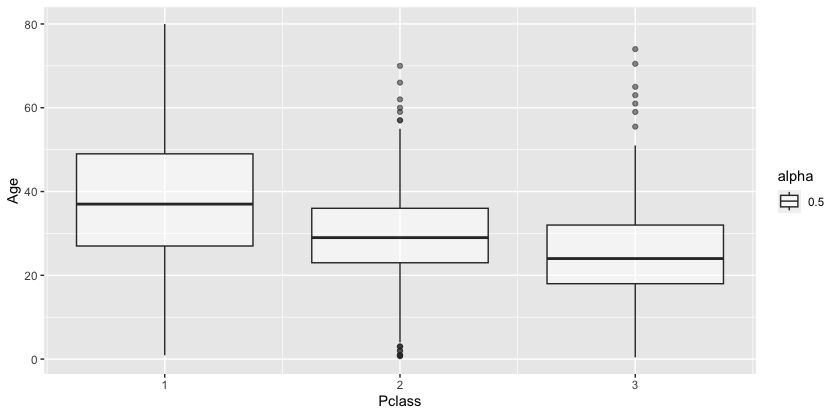
ggplot(datos,aes(Age))+geom\_histogram(bins=20,alpha=0.5,fill=’green’)



grafico<-ggplot(datos, aes(Pclass,Age))

grafico<-grafico+geom\_boxplot(aes(group=Pclass,alpha=0.5))

print(grafico) #grafico de bigotes

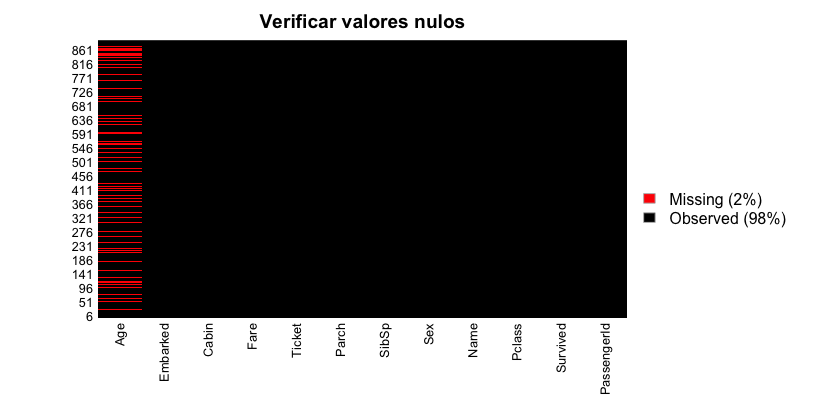


### 3.-Ejemplo de regresión logística – parte 2

install.packages(‘Amelia’)

library(Amelia)

missmap(datos, main='Verificar valores nulos',col=c('red','black'))



edad<-function(edad,clase){

salida<-edad

for(i in 1:length(edad)){

if(is.na(edad[i])){

if(clase[i]==1){

salida[i]<-38

}else if (clase[i]==2){

salida[i]<-29

}else{

salida[i]<-23

}

}else{

salida[i]<-edad[i]

}

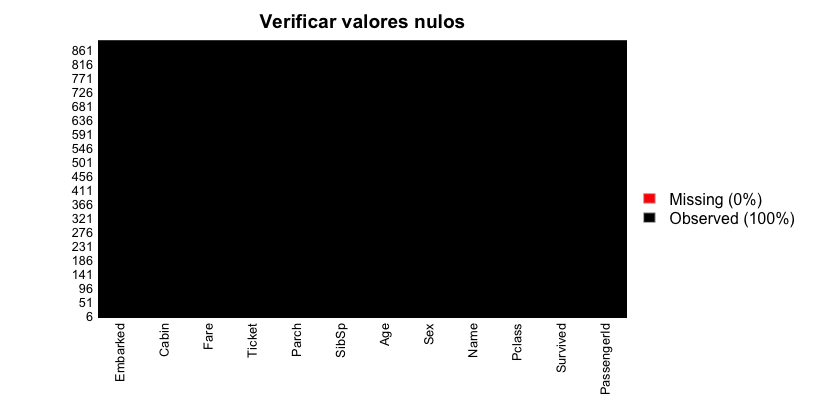
}

return(salida)

}

edades<-edad(datos$Age, datos$Pclass)

datos$Age<-edades

missmap(datos, main='Verificar valores nulos',col=c('red','black'))

### 4.-Ejemplo de regresión logística – parte 3

library(dplyr)

head(datos)

PassengerId Survived Pclass Name Sex Age SibSp Parch

1 1 0 3 Braund, Mr. Owen Harris male 22 1 0

2 2 1 1 Cumings, Mrs. John Bradley (Florence Briggs Thayer) female 38 1 0

3 3 1 3 Heikkinen, Miss. Laina female 26 0 0

4 4 1 1 Futrelle, Mrs. Jacques Heath (Lily May Peel) female 35 1 0

5 5 0 3 Allen, Mr. William Henry male 35 0 0

6 6 0 3 Moran, Mr. James male 23 0 0

Ticket Fare Cabin Embarked

1 A/5 21171 7.2500 S

2 PC 17599 71.2833 C85 C

3 STON/O2. 3101282 7.9250 S

4 113803 53.1000 C123 S

5 373450 8.0500 S

6 330877 8.4583 Q

datos<-select(datos, -PassengerId, -Name, -Ticket, -Cabin)

head(datos)

Survived Pclass Sex Age SibSp Parch Fare Embarked

1 0 3 male 22 1 0 7.2500 S

2 1 1 female 38 1 0 71.2833 C

3 1 3 female 26 0 0 7.9250 S

4 1 1 female 35 1 0 53.1000 S

5 0 3 male 35 0 0 8.0500 S

6 0 3 male 23 0 0 8.4583 Q

datos$Survived<-factor(datos$Survived)

datos$Pclass<-factor(datos$Pclass)

datos$Parch<-factor(datos$Parch)

datos$SibSp<-factor(datos$SibSp)

str(datos)

'data.frame': 891 obs. of 8 variables:

$ Survived: Factor w/ 2 levels "0","1": 1 2 2 2 1 1 1 1 2 2 ...

$ Pclass : Factor w/ 3 levels "1","2","3": 3 1 3 1 3 3 1 3 3 2 ...

$ Sex : chr "male" "female" "female" "female" ...

$ Age : num 22 38 26 35 35 23 54 2 27 14 ...

$ SibSp : Factor w/ 7 levels "0","1","2","3",..: 2 2 1 2 1 1 1 4 1 2 ...

$ Parch : Factor w/ 7 levels "0","1","2","3",..: 1 1 1 1 1 1 1 2 3 1 ...

$ Fare : num 7.25 71.28 7.92 53.1 8.05 ...

$ Embarked: chr "S" "C" "S" "S" ...

install.packages(‘caTools’)

library(caTools)

set.seed(90)

división <-sample.split(datos$Survived, SplitRatio=0.7)

entrenamiento <- subset(datos, división == TRUE)

pruebas <- subset(datos, división == FALSE)

head(entrenamiento)

Survived Pclass Sex Age SibSp Parch Fare Embarked

1 0 3 male 22 1 0 7.2500 S

3 1 3 female 26 0 0 7.9250 S

4 1 1 female 35 1 0 53.1000 S

6 0 3 male 23 0 0 8.4583 Q

8 0 3 male 2 3 1 21.0750 S

9 1 3 female 27 0 2 11.1333 S

head(pruebas)

Survived Pclass Sex Age SibSp Parch Fare Embarked

2 1 1 female 38 1 0 71.2833 C

5 0 3 male 35 0 0 8.0500 S

7 0 1 male 54 0 0 51.8625 S

11 1 3 female 4 1 1 16.7000 S

17 0 3 male 2 4 1 29.1250 Q

21 0 2 male 35 0 0 26.0000 S

modelo <-glm(Survived ~ . , family=binomial(link=’logit’), data=entrenamiento)

summary(modelo)

Call:

glm(formula = Survived ~ ., family = binomial(link = "logit"),

data = entrenamiento)

Deviance Residuals:

Min 1Q Median 3Q Max

-2.8977 -0.6047 -0.3882 0.5784 2.5421

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) 1.997e+01 3.956e+03 0.005 0.995972

Pclass2 -1.086e+00 3.667e-01 -2.962 0.003061 \*\*

Pclass3 -2.449e+00 3.806e-01 -6.434 1.24e-10 \*\*\*

Sexmale -2.607e+00 2.443e-01 -10.671 < 2e-16 \*\*\*

Age -4.071e-02 1.085e-02 -3.753 0.000175 \*\*\*

SibSp1 2.188e-01 2.700e-01 0.811 0.417594

SibSp2 -2.275e-02 6.501e-01 -0.035 0.972086

SibSp3 -2.191e+00 9.930e-01 -2.207 0.027347 \*

SibSp4 -1.226e+00 8.301e-01 -1.476 0.139815

SibSp5 -1.695e+01 1.742e+03 -0.010 0.992233

SibSp8 -1.701e+01 1.262e+03 -0.013 0.989242

Parch1 1.749e-01 3.429e-01 0.510 0.609958

Parch2 1.765e-01 4.680e-01 0.377 0.706041

Parch3 -2.175e-03 1.217e+00 -0.002 0.998573

Parch4 -1.717e+01 2.225e+03 -0.008 0.993846

Parch5 -1.668e+01 2.002e+03 -0.008 0.993352

Parch6 -1.730e+01 3.956e+03 -0.004 0.996511

Fare 1.483e-03 2.534e-03 0.585 0.558283

EmbarkedC -1.606e+01 3.956e+03 -0.004 0.996760

EmbarkedQ -1.560e+01 3.956e+03 -0.004 0.996854

EmbarkedS -1.633e+01 3.956e+03 -0.004 0.996707

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 829.60 on 622 degrees of freedom

Residual deviance: 523.19 on 602 degrees of freedom

AIC: 565.19

Number of Fisher Scoring iterations: 16

predicciones<-predict(modelo, pruebas, type=’response’)

predicciones

2 5 7 11 17 21 23 26

9.362195e-01 5.602931e-02 2.527096e-01 8.102256e-01 1.454883e-01 1.922965e-01 7.902907e-01 5.245491e-08

27 30 33 36 37 40 41 46

1.118158e-01 8.819434e-02 7.311676e-01 4.069475e-01 1.118164e-01 7.550800e-01 4.502060e-01 8.821274e-02

49 50 52 62 63 65 66 67

1.116859e-01 6.700059e-01 9.495353e-02 1.000000e+00 3.888723e-01 4.491642e-01 1.588673e-01 8.010910e-01

…………….

head (predicciones)

2 5 7 11 17 21

0.93621950 0.05602931 0.25270961 0.81022560 0.14548827 0.19229646

resultados <-ifelse(predicciones > 0.5, 1., 0)

head (resultados)

2 5 7 11 17 21

1. 0 0 1 0 0

head (datos)

Survived Pclass Sex Age SibSp Parch Fare Embarked

1 0 3 male 22 1 0 7.2500 S

2 1 1 female 38 1 0 71.2833 C

3 1 3 female 26 0 0 7.9250 S

4 1 1 female 35 1 0 53.1000 S

5 0 3 male 35 0 0 8.0500 S

6 0 3 male 23 0 0 8.4583 Q

error <- mean(resultados != pruebas$Survived)

error

[1] 0.1828358

Precisión <- error

precison <- 1 - precision

print(precison)

[1] 0.8171642

### **Sección 16. Machine Learning-K vecinos más cercanos**

### 1.-Introduccción a los k-vecinos más cercanos

El método de los k vecinos más cercanos es un método de clasificación supervisada, que estima la probabilidad de que un elemento ‘x’ pertenezca a una clase ‘C’ a partir de la información proporcionada.

En el reconocimiento de patrones, este algoritmo es usado como método de clasificación de objetos, basado en un entrenamiento mediante ejemplos cercanos en el espacio de los elementos.

Es tan sencillo, fácil de entrenar y trabaja con cualquier número de clases.

K vecinos más cercanos –KNN

1.-Calcula la distancia del círculo ‘verde’ a todos los demás elementos.

2.-Ordena los puntos en función de la distancia que los separa del verde, de menor a mayor.

3.-Predice el grupo al que pertenece el círculo ‘verde’ en función de la clase a la que pertenecen los ‘k’ elementos más cercanos.

### 2.-Ejemplo de los k-vecinos más cercanos – parte 1

install.packages(‘ISLR’)

library(ISLR)

datos<-Caravan

head(datos)

str(datos)

>

summary(datos$Purchase)

>

any(is.na(datos))

[1] FALSE

datos.compra<-datos[,86]

datos.estandarizados<-scale(datos[,-86])

filas<-1\_1000

head(filas)

pruebas.datos<-datos.estandarizados[filas,]

pruebas.compra<-datos.compra[filas]

entrenamiento.datos<-datos.estandarizados[-filas,]

entrenamiento.compra<-datos.compra[-filas]

### 3.-Ejemplo de los k-vecinos más cercanos – parte 2

library(class)

set.seed(90)

predicción.compra<-knn(entrenamiento.datos,pruebas.datos, entrenamiento.compra, k=1)

head(prediccion.compra)

error<-mean(pruebas.compra !=predicción.compra)

error

> [1] 0.116

predicción.compra<-knn(entrenamiento.datos,pruebas.datos, entrenamiento.compra, k=5)

error<-mean(pruebas.compra !=predicción.compra)

error

>[1] 0.066

redicción.compra<-NULL

errores<-NULL

for (i in 1:20){

set.seed(90)

predicción.compra<-knn(entrenamiento.datos,pruebas.datos, entrenamiento.compra, k=i)

errores[i]<-mean(pruebas.compra !=predicción.compra)

}

print(errores) #mejor for(i in 1:20) {print(errores[i])}

valores.k<-1:20

tabla.errores<-data.frame(errores, valores.k

tabla.errores

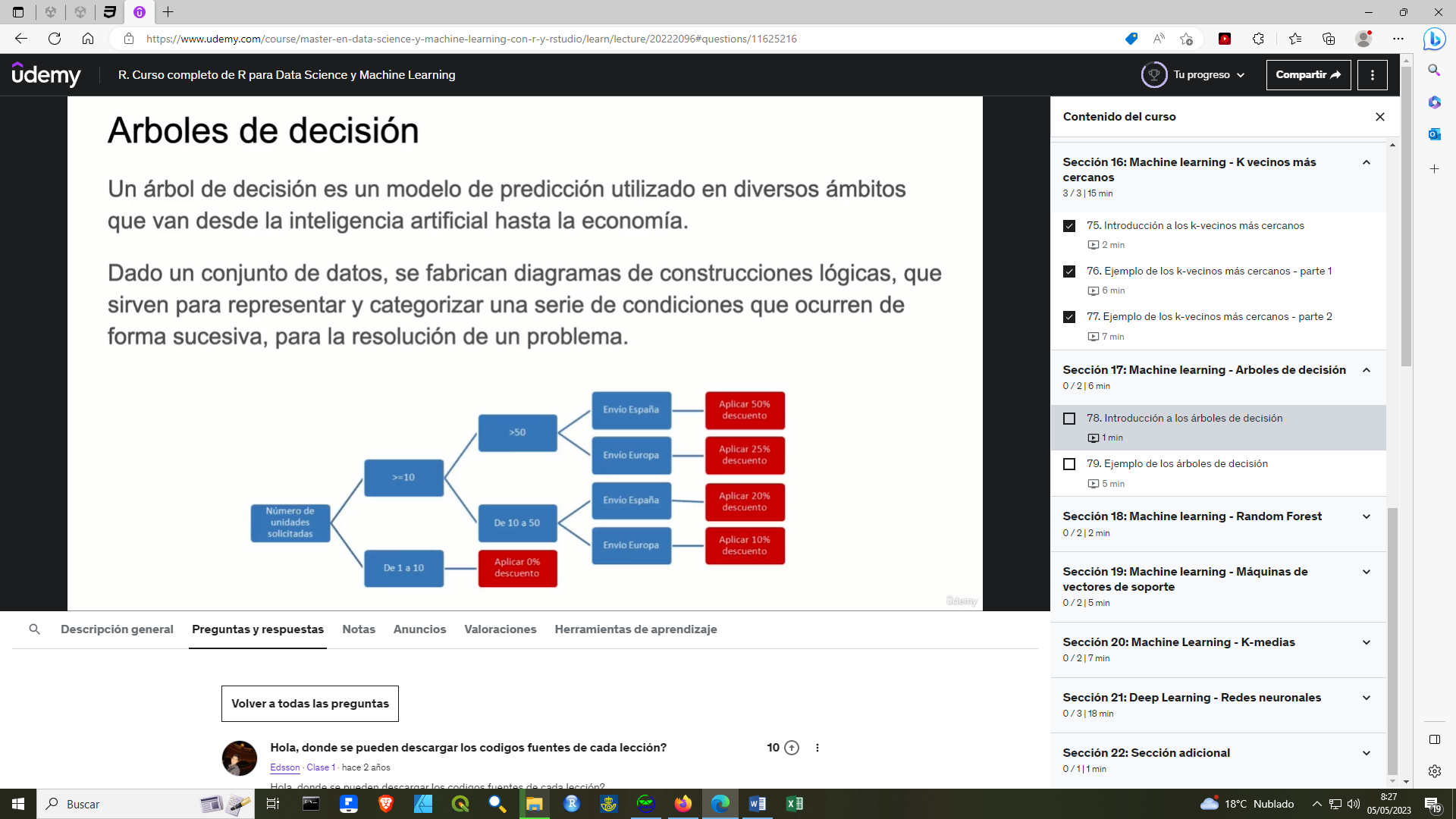
>

### **Sección 17. Machine Learning-Arboles de decisión**

### 1.-Introduccción a los árboles de decisión

Un árbol de decisión es un modelo de predicción utilizado en diversos ámbitos que van desde la inteligencia artificial hasta la economía.

Dado un conjunto de datos, se fabrican diagramas de construcciones lógicas, que sirven para representar y categorizar una serie de condiciones que ocurren de forma sucesiva, para la resolución de un problema.



install.packages(‘rpart’)

library(rpart)

datos<-kyphosis

str(datos)

'data.frame': 81 obs. of 4 variables:

$ Kyphosis: Factor w/ 2 levels "absent","present": 1 1 2 1 1 1 1 1 1 2 ...

$ Age : int 71 158 128 2 1 1 61 37 113 59 ...

$ Number : int 3 3 4 5 4 2 2 3 2 6 ...

$ Start : int 5 14 5 1 15 16 17 16 16 12 ...

arbol <- rpart(kyphosis,datos)

printcp(arbol)

Classification tree:

rpart(formula = kyphosis, data = datos)

Variables actually used in tree construction:

[1] Age Start

Root node error: 17/81 = 0.20988

n= 81

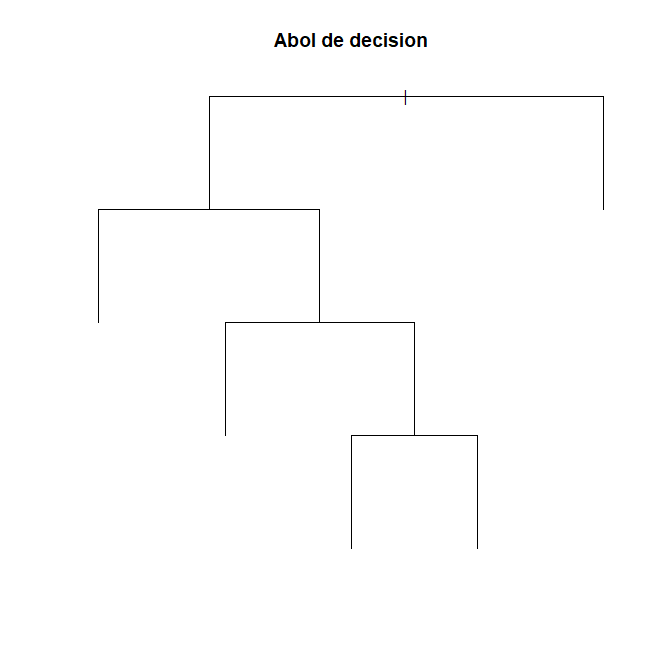
CP nsplit rel error xerror xstd

1 0.176471 0 1.00000 1.00000 0.21559

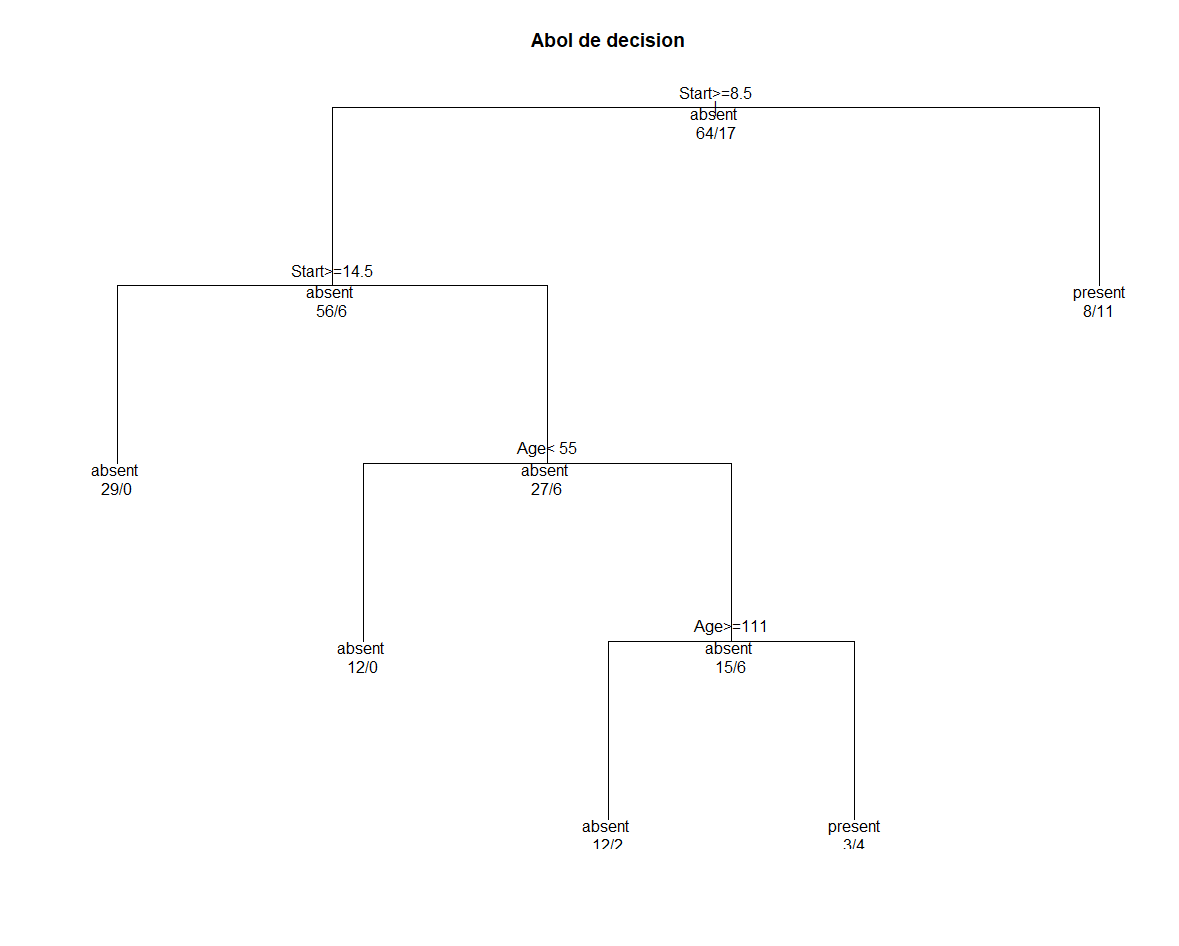
2 0.019608 1 0.82353 0.82353 0.20018

3 0.010000 4 0.76471 0.88235 0.20565

plot(arbol, uniform=TRUE, main='Abol de decision')



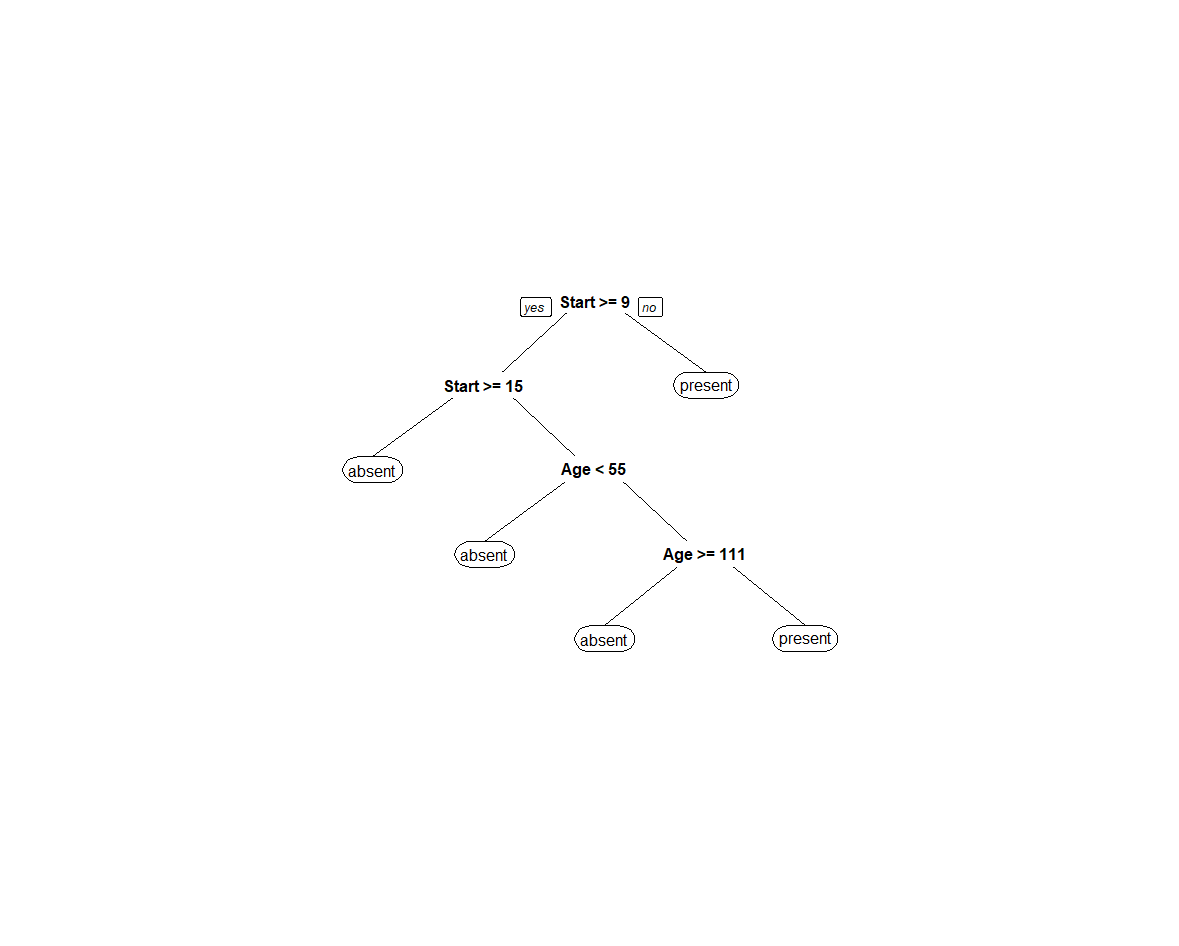
text(arbol, use.n=TRUE, all=TRUE)



install.packages('rpart.plot')

library(rpart.plot)

prp(arbol)



### 2.-Ejemplo de los árboles de decisión

library(rpart)

data(iris)

tree\_model <- rpart(Species ~ ., data = iris, method = "class")

printcp(tree\_model)

Classification tree:

rpart(formula = Species ~ ., data = iris, method = "class")

Variables actually used in tree construction:

[1] Petal.Length Petal.Width

Root node error: 100/150 = 0.66667

n= 150

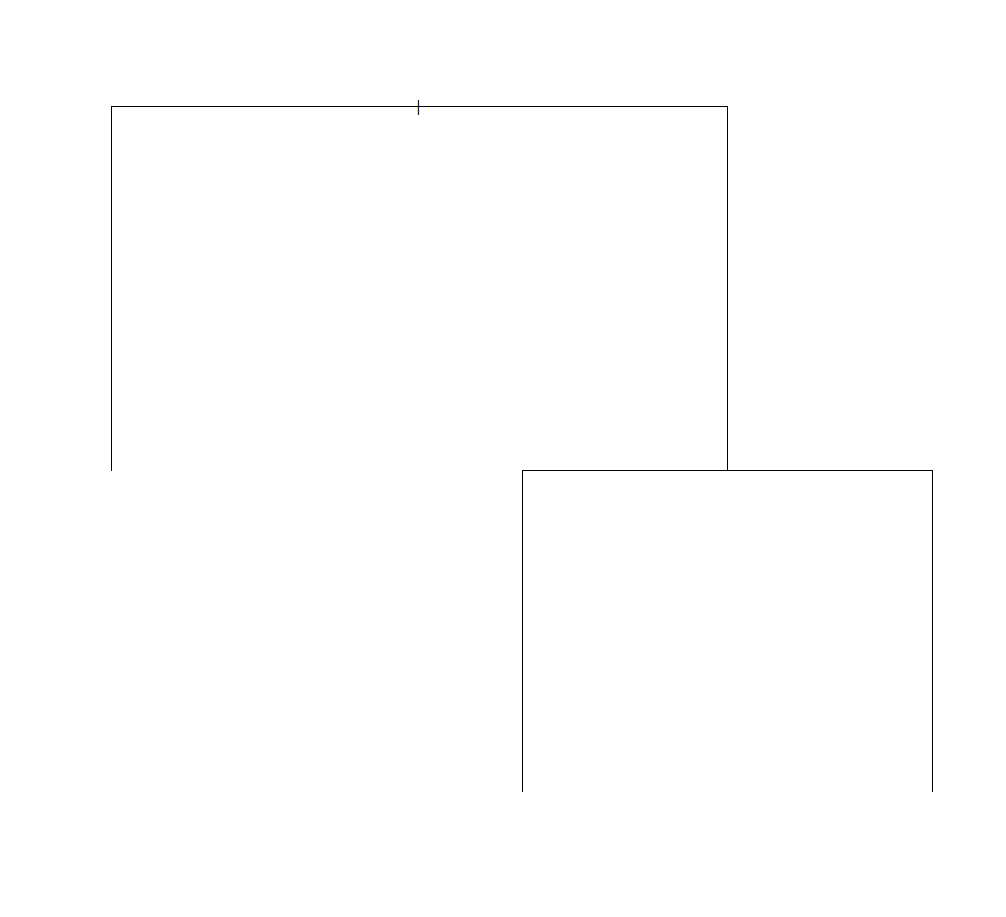
CP nsplit rel error xerror xstd

1 0.50 0 1.00 1.18 0.050173

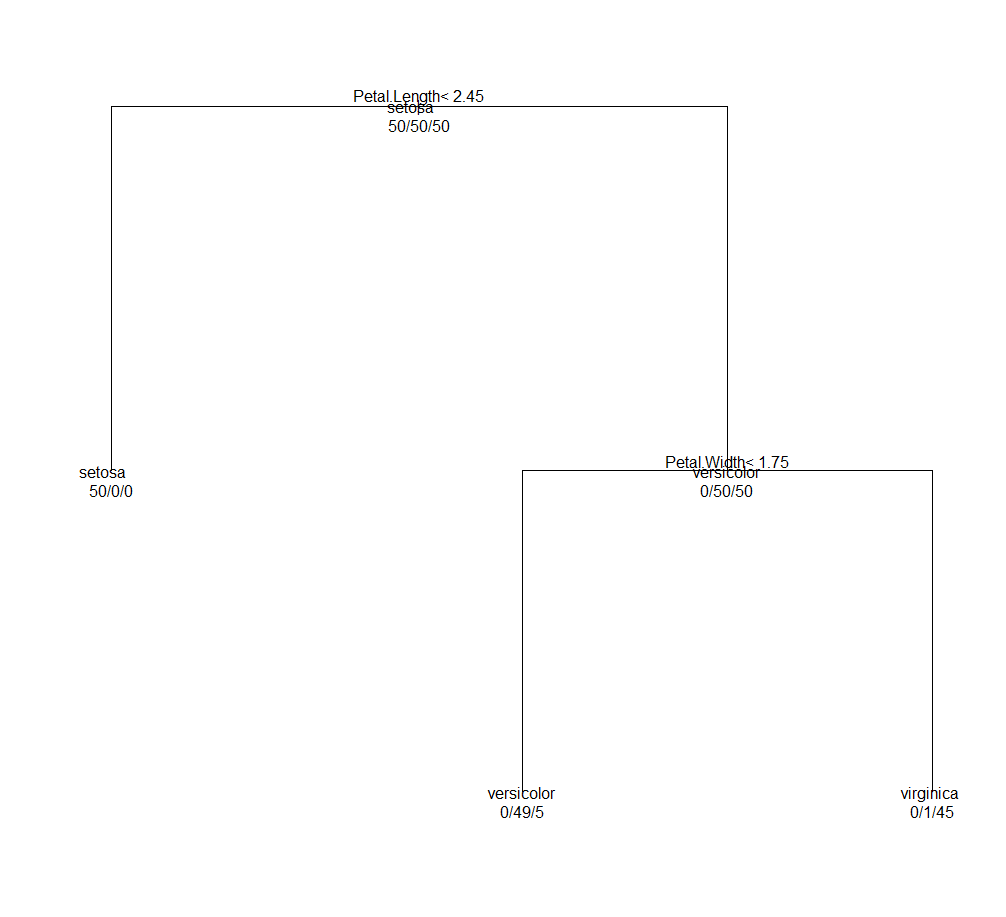
2 0.44 1 0.50 0.72 0.061188

3 0.01 2 0.06 0.08 0.027520

> plot(tree\_model)

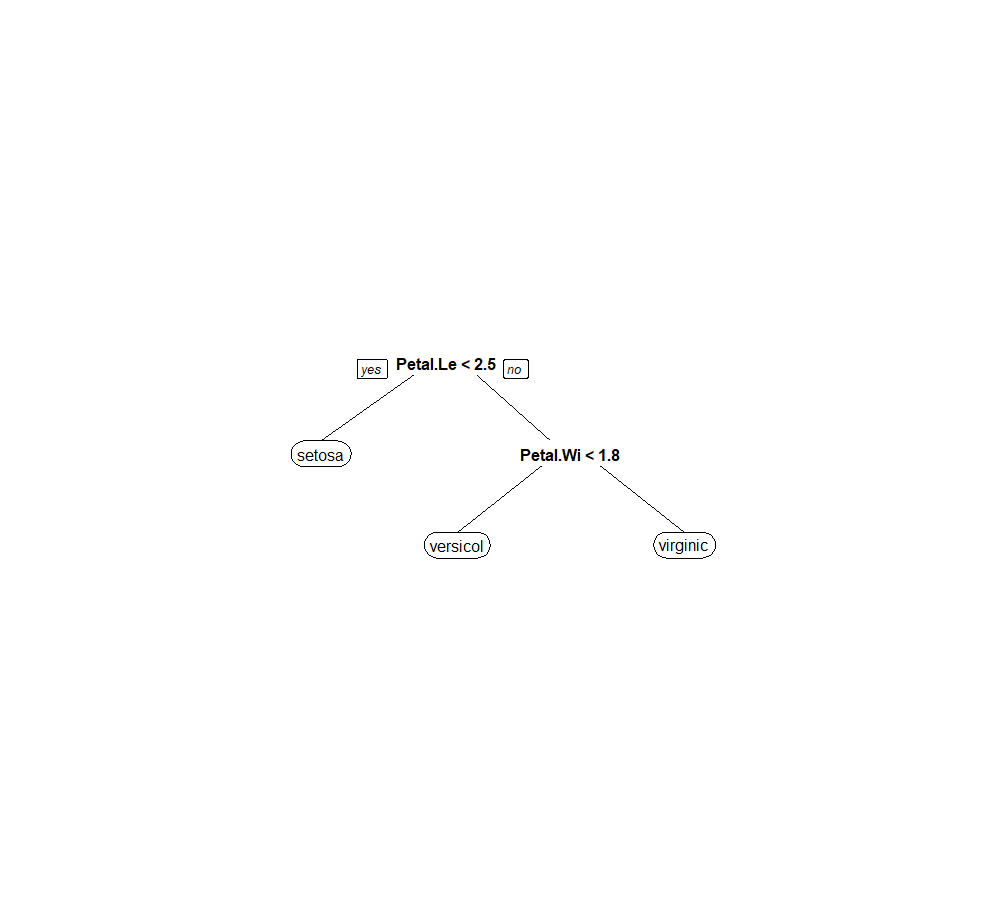


> text(tree\_model,use.n=TRUE,all=TRUE)



> library(rpart.plot

> prp(tree\_model)



### 3.-Otro ejemplo de los árboles de decisión

titanic <- read.csv("titanic.csv")

titanic$Survived <- factor(titanic$Survived)

> head(titanic$Survived)

[1] 0 1 1 1 0 0

Levels: 0 1

> summary(titanic$Survived)

0 1

549 342

titanic <- titanic[, c("Survived", "Pclass", "Sex", "Age", "Fare")]

> summary(titanic)

Survived Pclass Sex Age Fare

0:549 Min. :1.000 Length:891 Min. : 0.42 Min. : 0.00

1:342 1st Qu.:2.000 Class :character 1st Qu.:20.12 1st Qu.: 7.91

Median :3.000 Mode :character Median :28.00 Median : 14.45

Mean :2.309 Mean :29.70 Mean : 32.20

3rd Qu.:3.000 3rd Qu.:38.00 3rd Qu.: 31.00

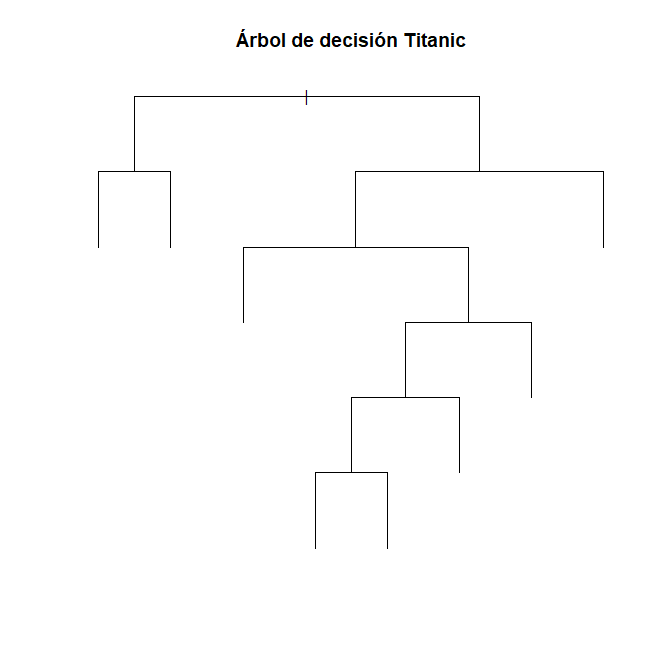
Max. :3.000 Max. :80.00 Max. :512.33

NA's :177

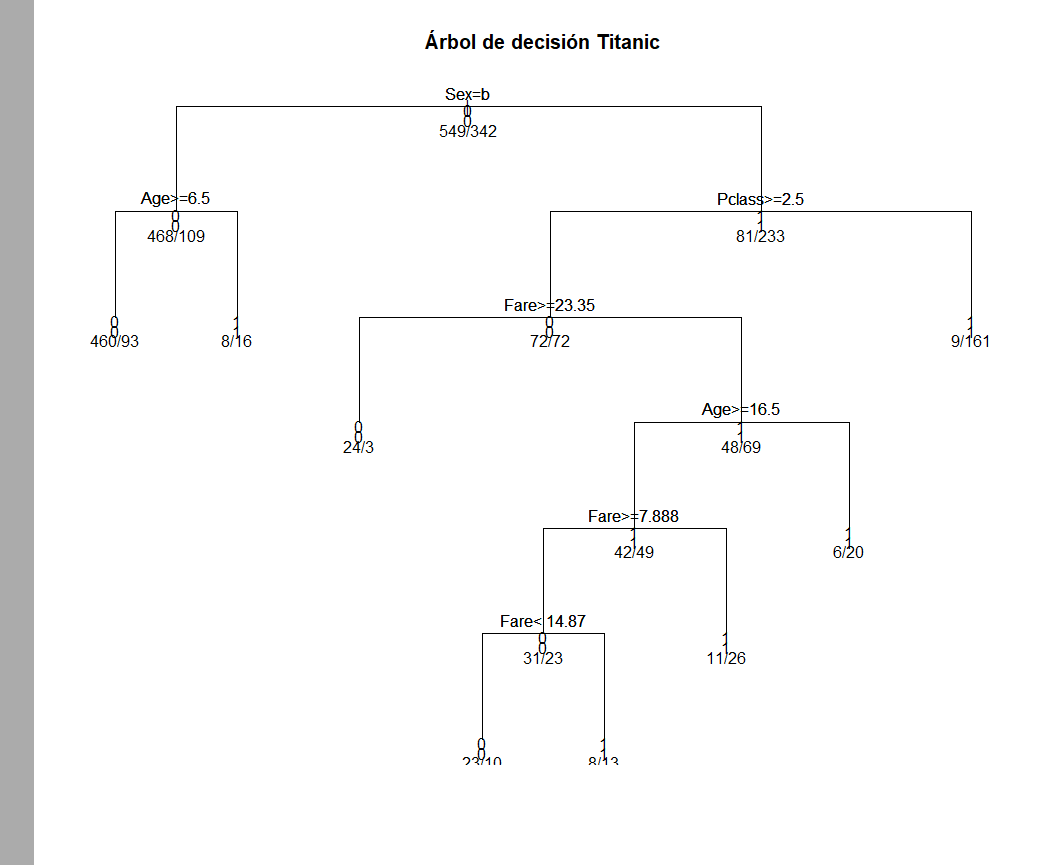
library(rpart)

tree\_model <- rpart(Survived ~ Pclass + Sex + Age + Fare, data = titanic, method = "class")

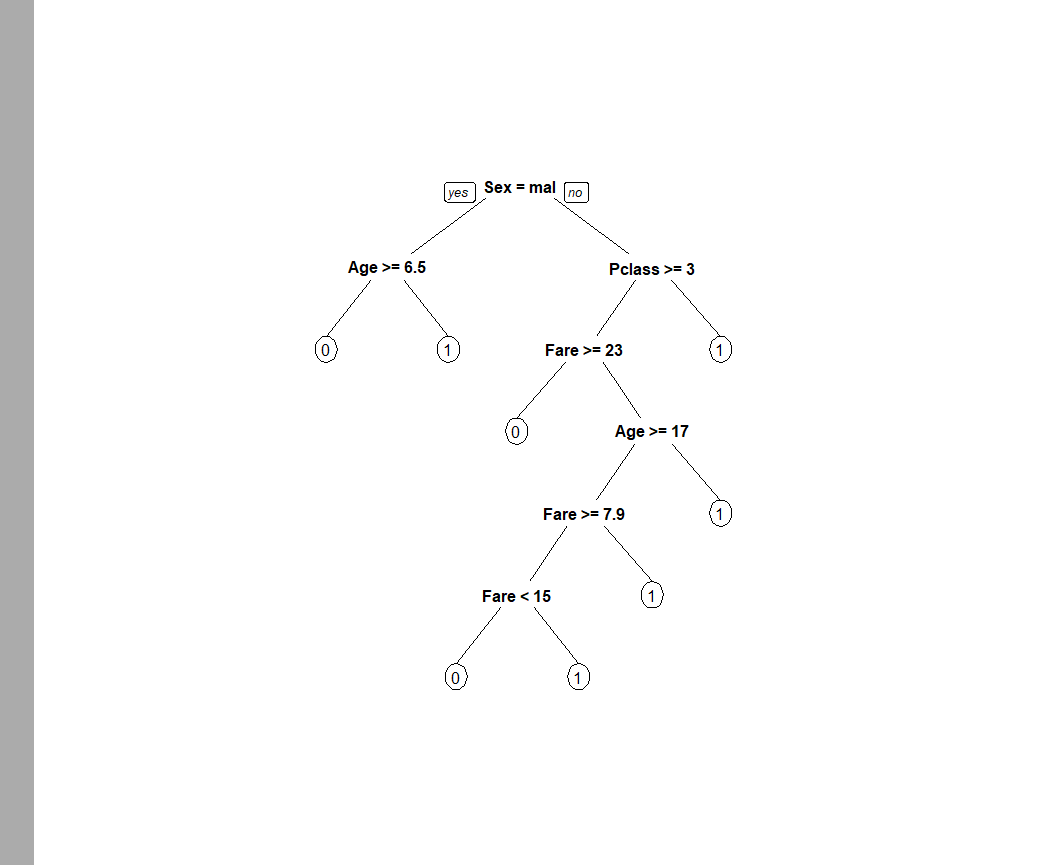
> plot(tree\_model, uniform=TRUE, main='Árbol de decisión Titanic')



> text(tree\_model, use.n=TRUE, all=TRUE)



prp(tree\_model)



### **Sección 18. Machine Learning-Random Forest**

### 1.-Introduccción a Random Forest

### 2.-Ejemplo de Random Forest

### **Sección 19. Machine Learning-Máquinas de vetores de soporte**

### 1.-Introduccción a las máquinas de vectores de soporte

### 2.-Ejemplo de máquinas de vectores de soporte

### **Sección 20. Machine Learning- K - medias**

### 1.-Introduccción a k-medias

### 2.-Ejemplo de k-medias

### **Sección 21. Deep Learning – Redes neuronales**

### 1.-Introduccción a redes neuronales

### 2.-Ejemplo de red neuronal – parte 1

### 3.-Ejemplo de red neuronal – parte 2

### **Sección 22. Sección adicional**

### 1.-Clase extra