# Reto.4 - Practica 2: Limpieza y análisis de datos

M2.851 – Tipología y ciclo de vida de los datos.

Alberto Suárez y Leonor Pallarés

9 de enero 2024

## Contents

Partimos de dataset **heart.csv**, disponible en Kaggle **https://www.kaggle.com**, este dataset contiene información referente a pacientes con enfermedades cardiacas.

El objetivo de esta actividad será limpiar, normalizar si es necesario y establecer visualizaciones que nos permitan obtener infomracion necesaria para poder establecer una predicion y análisis de ataques cardiacos.

# 1 Descripción del dataset.

**¿Por qué es importante y qué pregunta/problema pretende responder?**

Este dataset es interesante a nivel médico ya que se obtiene una clasificacion de ataques cardiacos de forma que se pueda llegar a diagnosticar la gravedad de un ataque en base a la información obtenida de numerosos pacientes. Este dataset es el propuesto en el enunciado de la Practica 2, se llama "Heart Attack Analysis & Prediction dataset" y puede encontrarse en la página web de Kaggle en la siguiente URL:<https://www.kaggle.com/datasets/rashikrahmanpritom/heart-attack-analysis-prediction-dataset>.

Este conjunto de datos contiene un total de 303 observaciones (los cuales representan a informacion sobre 303 pacientes diferentes) y 14 variables (que incluye información como edad, sexo, tipo de dolor en el pecho, presión arterial en reposo, niveles de colesterol, niveles de azúcar en sangre en ayunas, resultados electrocardiográficos en reposo, frecuencia cardíaca máxima alcanzada, angina inducida por el ejercicio, oldpeak (depresión del ST inducida por el ejercicio en relación con el reposo), pendiente del segmento ST de ejercicio máximo, número de vasos principales coloreados por fluoroscopia, talasemia (tipo de trastorno sanguíneo) y presencia o ausencia de enfermedad cardíaca).

Descripcion de las caracteristicas:

**01. age** - Edad en años
**02. sex** - Sexo (1 = masculino; 0 = femenino
**03. cp** - Tipo de dolor en el pecho (1 = angina típica; 2 = angina atípica; 3 = dolor no anginoso; 0 = asintomático)
**04. trestbps** - presión arterial en reposo (en mm Hg al ingresar al hospital)
**05. chol** - Colesterol sérico en mg/dl
**06. fbs** - Azúcar en sangre en ayunas > 120 mg/dl (1 = verdadero; 0 = falso)
**07. restecg** - Resultados electrocardiográficos en reposo (1 = normal; 2 = con anomalía de la onda ST-T; 0 = hipertrofia)
**08. thalach** - Frecuencia cardíaca máxima alcanzada
**09. exang** - Angina inducida por el ejercicio (1 = sí; 0 = no)
**10. oldpeak** - Depresión del ST inducida por el ejercicio en relación con el reposo
**11. slp** - Pendiente del segmento ST del ejercicio máximo (2 = ascendente; 1 = plano; 0 = descendente)
**12. caa** - Número de vasos principales (0-3) coloreados por fluoroscopia
**13. thall** - Tasa de hemoglobina (talasemia) (1 = defecto solucionado; 2 = normal; 3 = defecto reversible)
**14.output** - Atributo Objetivo o previsto de diagnóstico de enfermedad cardíaca (Valor 0 = < estrechamiento del diámetro (menos posibilidades de sufrir un ataque cardíaco); Valor 1 = > 50 % de estrechamiento del diámetro (menos posibilidades de sufrir un ataque cardíaco))

El dataset y el código del mismo se puede encontrar en la siguiente ubicación: https://github.com/pallaresl/PRA2_ASR_LPO

Si se consultan los datos de origen y se realiza alguna visualizacion a simple vista, se observa que el num. entre hombres (valor 1) y mujeres (valor 0) es muy dispar, existe mas información sobre hombres (207 obs) frente a mujeres (96 obs), lo cual representa que el 68% de la muestra son hombres y por tanto destaca que estos son más propensos a sufrir ataques cardiacos que los mujeres

En esta práctica limpiaremos los datos y trataremos de estimar un modelo que a partir de ellos pueda predecir la tipologia del ataque y sobre todo en que grado (clasificacion) teniendo en cuenta los diferentes parametros descritos en el dataset.

# 2 Integracion y Selección

**Integración y selección de los datos de interés a analizar. Puede ser el resultado de adicionar diferentes datasets o una subselección útil de los datos originales, en base al objetivo que se quiera conseguir**

Este conjunto de datos, consta de 2 ficheros "heart.csv" y "o2Saturation.csv", por lo que la idea inicial era realizar una fusión de ambos, pero después de observar y analizar ambos ficheros vemos que no existe una relacion de correspondencia real entre ambos (heart - o2Saturation) ni por num. de registros, ni campos clave, por tanto descartamos la integración de ambos ficheros. Asi pues, el análisis se realizará tras la importación del conjunto de datos **heart.csv**.

Con la lectura del fichero con la función **read.csv()** se consigue importar los datos en el data frame "heart" para su analisis, como se ve acontinuacion. .

```
##
# Lectura de dataset Heart.csv y comprobación de datos de lectura
#
heart <- read.csv("heart.csv")
n.var <- names(heart)
head(heart,5)
```

```
##   age sex cp trtbps chol fbs restecg thalachh exng oldpeak slp caa thall output
## 1  63   1  3    145  233   1       0      150    0     2.3   0   0     1      1
## 2  37   1  2    130  250   0       1      187    0     3.5   0   0     2      1
## 3  41   0  1    130  204   0       0      172    0     1.4   2   0     2      1
## 4  56   1  1    120  236   0       1      178    0     0.8   2   0     2      1
## 5  57   0  0    120  354   0       1      163    1     0.6   2   0     2      1
```

```
#
##
```

En un primer análisis, los datos de interes que se desprenden de forma generica son que el fichero de datos contiene 303 registros y 14 variables. de los cuales 96 son mujeres y 207 hombres.

Las variables que lo componen y el formato de estas son las siguientes:

```
# Obtenemos datos básicos variables y formatos
res <- sapply(heart,class)
kable(data.frame(Var=names(res),Tipo=as.vector(res)))
```

| Var | Tipo |
|---|---|
| age | integer |
| sex | integer |
| cp | integer |
| trtbps | integer |
| chol | integer |
| fbs | integer |
| restecg | integer |
| thalachh | integer |
| exng | integer |
| oldpeak | numeric |
| slp | integer |
| caa | integer |
| thall | integer |
| output | integer |

```
#
##
```

De todas las variables, hay muchas que son categoricas, por tanto pueden normalizarse de int a factorial, si aplicamos esto quedará el dataset de la siguiente manera:

```
##
#Obtenemos datos básicos, con la conversion a factores
heart$sex     <- factor(heart$sex)
heart$cp      <- factor(heart$cp)
heart$fbs     <- factor(heart$fbs)
heart$restecg <- factor(heart$restecg)
heart$exng    <- factor(heart$exng)
heart$slp     <- factor(heart$slp)
heart$caa     <- factor(heart$caa)
heart$thall   <- factor(heart$thall)
heart$output  <- factor(heart$output)
#
#Mostramos los datos
str(heart)
```

```
## 'data.frame':    303 obs. of  14 variables:
##  $ age     : int  63 37 41 56 57 57 56 44 52 57 ...
##  $ sex     : Factor w/ 2 levels "0","1": 2 2 1 2 1 2 1 2 2 2 ...
##  $ cp      : Factor w/ 4 levels "0","1","2","3": 4 3 2 2 1 1 2 2 3 3 ...
##  $ trtbps  : int  145 130 130 120 120 140 140 120 172 150 ...
##  $ chol    : int  233 250 204 236 354 192 294 263 199 168 ...
##  $ fbs     : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 1 2 1 ...
##  $ restecg : Factor w/ 3 levels "0","1","2": 1 2 1 2 2 2 1 2 2 2 ...
##  $ thalachh: int  150 187 172 178 163 148 153 173 162 174 ...
##  $ exng    : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 1 1 1 ...
##  $ oldpeak : num  2.3 3.5 1.4 0.8 0.6 0.4 1.3 0 0.5 1.6 ...
##  $ slp     : Factor w/ 3 levels "0","1","2": 1 1 3 3 3 2 2 3 3 3 ...
##  $ caa     : Factor w/ 5 levels "0","1","2","3",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ thall   : Factor w/ 4 levels "0","1","2","3": 2 3 3 3 3 2 3 4 4 3 ...
##  $ output  : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
```

```
#
##
```

Ahora con los formatos de las variables revisadas procedemos a realizar un Analisis básico de los datos de todo el dataset, para ello utilizaremos la funcion **summary()**

```
##
#Analisis basicos de los datos (estadisticas)
summary(heart)
```

```
##       age        sex     cp          trtbps           chol       fbs
##  Min.   :29.00   0: 96   0:143   Min.   : 94.0   Min.   :126.0   0:258
##  1st Qu.:47.50   1:207   1: 50   1st Qu.:120.0   1st Qu.:211.0   1: 45
##  Median :55.00           2: 87   Median :130.0   Median :240.0
##  Mean   :54.37           3: 23   Mean   :131.6   Mean   :246.3
##  3rd Qu.:61.00                   3rd Qu.:140.0   3rd Qu.:274.5
##  Max.   :77.00                   Max.   :200.0   Max.   :564.0
##  restecg    thalachh     exng       oldpeak     slp     caa     thall   output
##  0:147   Min.   : 71.0   0:204   Min.   :0.00   0: 21   0:175   0:  2   0:138
##  1:152   1st Qu.:133.5   1: 99   1st Qu.:0.00   1:140   1: 65   1: 18   1:165
```

```
##  2:  4   Median :153.0          Median :0.80   2:142   2: 38   2:166
##          Mean   :149.6          Mean   :1.04           3: 20   3:117
##          3rd Qu.:166.0          3rd Qu.:1.60           4:  5
##          Max.   :202.0          Max.   :6.20
```

```
#
##
```

Con los resultados obtenidos, se ha consideramos que no vamos a recoger un subconjunto de datos, ya que todas las variables pueden ser concluyentes de cara a identificar (clasificar) o pronosticar un posible ataque cardiaco, tanto por sexo, edad o por sexo-edad.

# 3 Limpieza de datos

## 3.1 ¿Los datos contienen ceros o elementos vacíos? Gestiona cada uno de estos casos.

En este apartado se realizarán dos comprobaciones, en primer lugar se comprobará si existe algún valor perdido (*NA*) en el dataset, y en segundo lugar se analizará la variable **thall**, ya que en el resúmen del analisis de datos básicos generado anteriormente se observa que existen valores **0**, por lo tanto se analizará si este 0 es un valor correcto o si es un valor perdido que ha sido rellenado por defecto a 0.

En primer lugar, se analiza si existen **valores perdidos** en el dataset mediante la función *is.na* de R:

```
##
#Preguntamos si hay algun valor perdido en todo el dataset
sapply(heart, function(x)(sum(is.na(x)))) # NA counts
```

```
##      age      sex       cp   trtbps     chol      fbs  restecg thalachh
##        0        0        0        0        0        0        0        0
##     exng  oldpeak      slp      caa    thall   output
##        0        0        0        0        0        0
```

```
#
##
```

por el resultado obtenido comprobamos que no existe ningún valor nulo en el conjunto de datos, en el supuesto caso de que hubieramos encontrado algun "NA", se analizaria la distribucion de este y se plantearia la opcion de reemplazarlos por un estadístico (la media, mediana..) o en el caso de que el numero de NA de alguna variable fuera muy elevado se plantearia descartar esa variable en concreto del dataset.

A continuacion, se analiza con un poco mas de detalle la variable **thall**, esto se debe a que según la definición de nuestro dataset, el rango de valores permitido para esta variable está entre 1 y el 3, sin embargo en el analisis de datos basicos previo, se detectan que hay 2 con valor a 0 por lo que se deduce que existen valores perdidos en esta variable, quizas debido a un error de transcripcion.

Los registros en los que detectamos 0 en esta variable son:

```
##
# Obtenceion de registros con thall = 0
head(heart[heart$thall==0,])
```

```
##     age sex cp trtbps chol fbs restecg thalachh exng oldpeak slp caa thall
## 49   53   0  2    128  216   0       0      115    0       0   2   0     0
## 282  52   1  0    128  204   1       1      156    1       1   1   0     0
##     output
## 49       1
## 282      0
```

```
#
#Calculmos la moda: La frecuencia de valores de la variable
tf_thall <- table(heart$thall)
moda <- as.numeric(names(tf_thall)[which.max(tf_thall)])
#
##
```

y como observamos que hay pocos valores, se toma la decision de proceder a sustituilo por el valor de la **moda** = 2 calculada en el chunk anterior.

```
##
#Substituimos los valors 0 por la moda = 2, volvemos a factorizar para reorganizar los niveles
heart$thall   <- replace(heart$thall, heart$thall==0, 2)
heart$thall   <- factor(heart$thall)
#
##
```

Si ahora volvemos a mostrar el detalle de esta variable podemos ver que el factorial ya no contempla el valor 0, como nivel factorial.

```
##
str(heart$thall)
```

```
##  Factor w/ 3 levels "1","2","3": 1 2 2 2 2 1 2 3 3 2 ...
```

```
#
##
```

## 3.2 Identifica y gestiona los valores extremos

En este apartado del tercer ejercicio de la práctica se analizarán e identificarán los valores extremos de las variables numéricas del dataset. Para ello se utilizará la función *boxplot.stats()* de R, la cual mostrará para cada variable, sus valores extremos:

```
## Filtrar solo las variables numéricas continuas
varNUM <- sapply(heart, is.numeric) & sapply(heart, function(x) length(unique(x)) > 5)
datos_numericos <- heart[, varNUM]
#
# Obtenemos los outilers de las variables numericas
boxplot.stats(heart$age)$out
```

```
## integer(0)
```

```
boxplot.stats(heart$trtbps)$out
```

```
## [1] 172 178 180 180 200 174 192 178 180
```

```
boxplot.stats(heart$chol)$out
```

```
## [1] 417 564 394 407 409
```

```
boxplot.stats(heart$thalachh)$out
```

```
## [1] 71
```

```
boxplot.stats(heart$oldpeak)$out
```

```
## [1] 4.2 6.2 5.6 4.2 4.4
```

```
#
## Las medias de estas variables estan
```

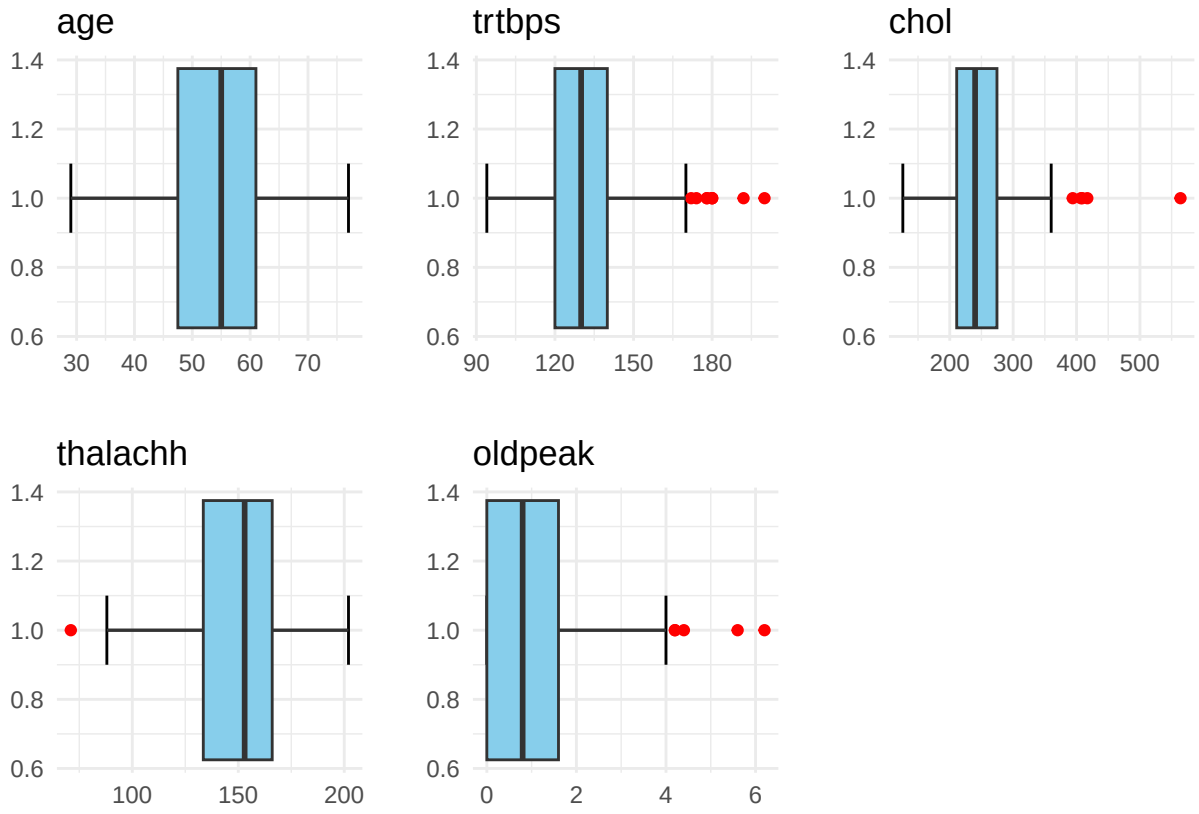
```
#summary(datos_numericos)
#
##
```

Si vemos una representacion grafica teniendo en cuenta los outliers, tendriamos

```
##
# Configurar el tamaño de la imagen
options(repr.plot.width = 12, repr.plot.height = 6)

# Crear boxplots para variables numéricas continuas y almacenarlos en una lista
plots <- lapply(colnames(datos_numericos), function(columna) {
  ggplot(heart, aes(x = 1, y = heart[[columna]])) +
    stat_boxplot(geom = "errorbar", width = 0.2)        +    # Bigotes
    geom_boxplot(fill = "skyblue",outlier.colour="red") +
    ggtitle(columna) +
    theme_minimal() +
    coord_flip() +
    ylab("") +
    xlab("")
})

# Visualizar la lista de boxplots
gridExtra::grid.arrange(grobs = plots, ncol = 3) #Se puede ajustar el núm de columnas según necesidad
```




```
#
# Restaurar la configuración del tamaño de la imagen
```

```r
options(repr.plot.width = 7, repr.plot.height = 5)
```

Como podemos observar, salvo la variable **age**, las demas presentan Outliers, valores extremos representados por los puntos rojos fuera de los "bigotes" de las cajas. Estos casos, los trataremos en los apartados siguientes para evitar distorsiones en el modelo de datos.

# 4 Análisis de los datos

## 4.1 Selección de los grupos de datos que se quieren analizar/comparar

El conjunto de datos seleccionado, a priori contiene informacion relevante en todos sus campos, ya que no existe ningun campo como identificador único por registro ni vemos a simple vista campo alguno con informacion innecesaia para el estudio que permita evaluar/clasificar el tipo de riesgo(alto/bajo) en el que se encuentra un paciente, a través de la variable output.

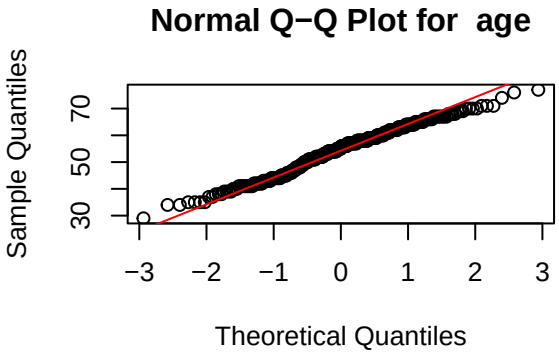
Los análisis a aplicar en esta práctica son el cálculo de la correlación entre las variables numericas del conjunto de datos y la variable dependiente "output". un contraste de hipótesis para comprobar cuales son los valores de las variables que más correlacionadas estan con la variable dependiente "output" y si difieren para las diferentes casuisticas, y en último lugar, se establecerá un modelo para asegurar cuáles son las variables que más afectan a la calidad del paciente.

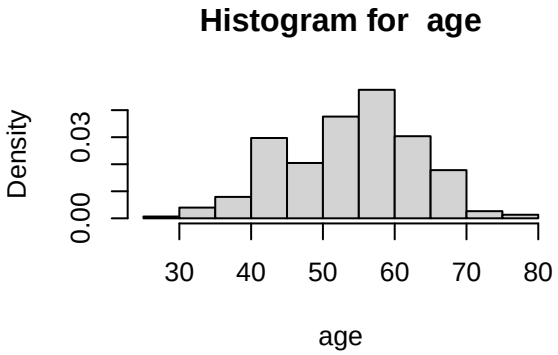
## 4.2 Comprobación de la normalidad y homogeneidad de la varianza.

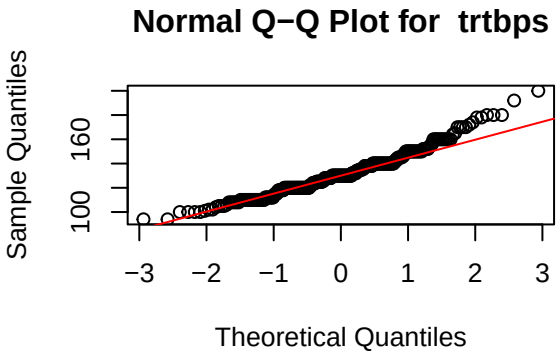
Normalidad

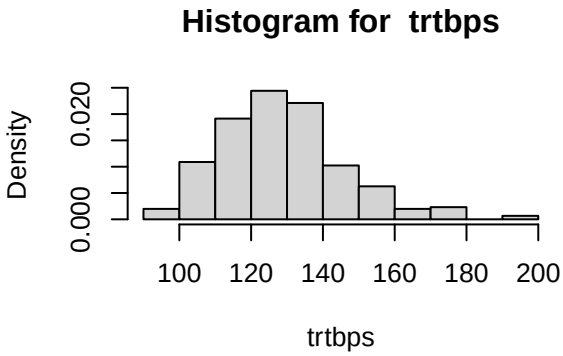
Para revisar si las variables pueden ser candidatas a la normalización miramos las graficas de quantile-quantile plot y el histograma.

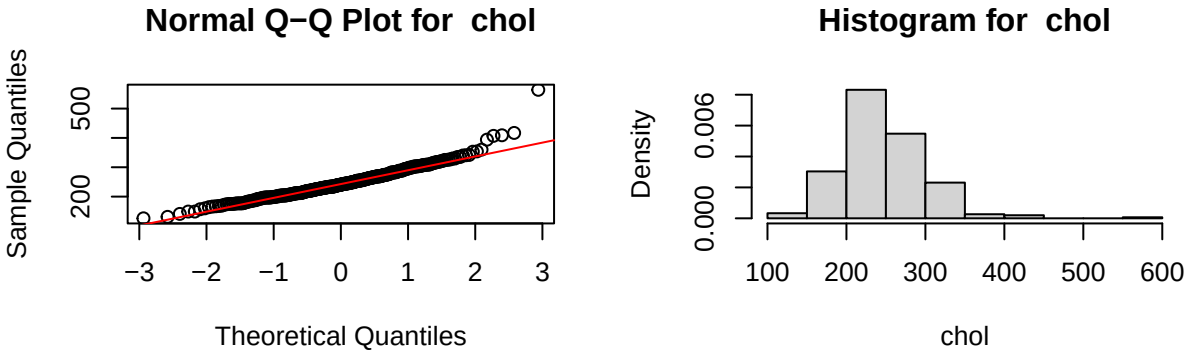
```r
##
#Generar graficos e histogramas
par(mfrow=c(2,2))
for(i in 1:ncol(datos_numericos)) {
  if (is.numeric(datos_numericos[,i])){
     qqnorm(datos_numericos[,i],main = paste("Normal Q-Q Plot for ",
                                colnames(datos_numericos)[i]))
     qqline(datos_numericos[,i],col="red")
     hist(datos_numericos[,i],main=paste("Histogram for ",
                                colnames(datos_numericos)[i]),
                                xlab=colnames(datos_numericos)[i],
                                freq = FALSE
          )
  }
}
```


Normal Q−Q Plot for age
Sample Quantiles
30
50
70
−3
−2
−1
0
1
2
3
Theoretical Quantiles

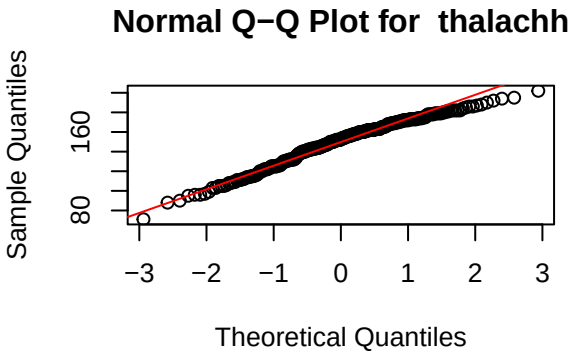

Histogram for age
Density
0.00
0.03
30
40
50
60
70
80
age

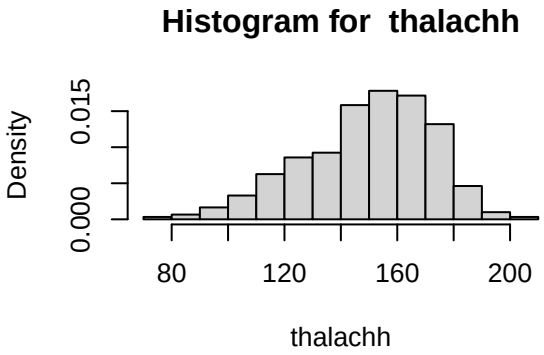

Normal Q−Q Plot for trtbps
Sample Quantiles
100
160
−3
−2
−1
0
1
2
3
Theoretical Quantiles


Histogram for trtbps
Density
0.000
0.020
100
120
140
160
180
200
trtbps

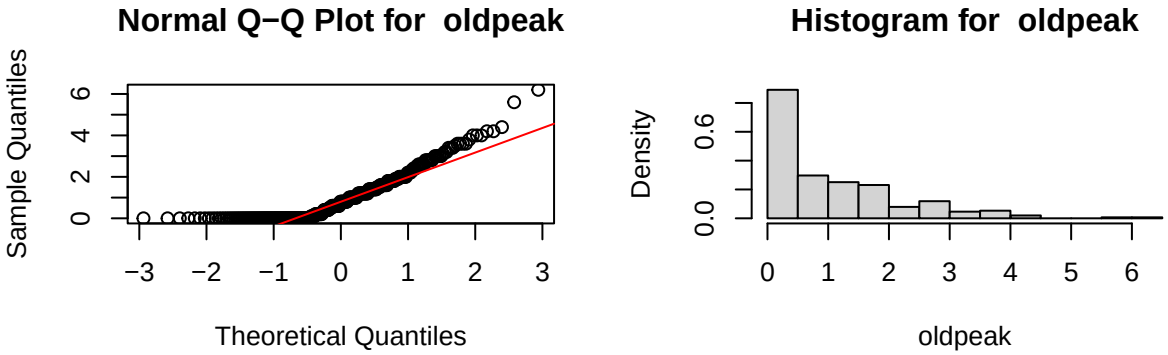

Normal Q−Q Plot for chol
Sample Quantiles
200
500
−3
−2
−1
0
1
2
3
Theoretical Quantiles


Histogram for chol
Density
0.000
0.006
100
200
300
400
500
600
chol


Normal Q−Q Plot for thalachh
Sample Quantiles
80
160
−3
−2
−1
0
1
2
3
Theoretical Quantiles


Histogram for thalachh
Density
0.000
0.015
80
120
160
200
thalachh

```
#
##
```




Los resultados del quantile-quantile plot nos indica que las variables pueden ser candidatas a la normalización en caso de ser necesario.

Para poder comprobar la normalidad de las variables cuantitativas del conjunto de datos se realizará el **test de Shapiro** a cada una de ellas y se mostrará por pantalla que variables siguen distribucion normal,cuales no y sus valores.

con este test, resolveremos la hpótesis nula con un nivel de confianza del 95%, para resolver si la poblacion del dataset está distribuida normalmente.

Hipotesis Nula ($H_0$): se cumplira si el p-value es menor a alfa (nivel de significancia =0,5). Distribuida normalmente Hipotesis Alternativa($H_1$): se cumplira si el p-value es mayor o igual a alfa (nivel de significancia =0,5), rechazo de $H_0$, por lo que concluiremos que los datos no provienens de una distribucion normal.

```
##
# Para un nivel de significancia (alpha) = 0.05
alpha <- 0.05
vars  <- colnames(datos_numericos) #Variables cuantitativas
v_Norm   <- c()
v_noNorm <- c()

for(i in 1:length(vars)){
  if(shapiro.test(datos_numericos[,i])$p.value < alpha){
    v_noNorm <- c(v_noNorm,vars[i])
  } else {
    v_Norm <- c(v_Norm,vars[i])
  }
}
```

```r
shapiro.test(datos_numericos$age)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  datos_numericos$age
## W = 0.98637, p-value = 0.005798
```

```r
shapiro.test(datos_numericos$trtbp)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  datos_numericos$trtbp
## W = 0.96592, p-value = 1.458e-06
```

```r
shapiro.test(datos_numericos$chol)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  datos_numericos$chol
## W = 0.94688, p-value = 5.365e-09
```

```r
shapiro.test(datos_numericos$thalachh)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  datos_numericos$thalachh
## W = 0.97632, p-value = 6.621e-05
```

```r
shapiro.test(datos_numericos$oldpeak)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  datos_numericos$oldpeak
## W = 0.84418, p-value < 2.2e-16
```

```r
#
##
```

En base a los test realizados las variables que siguen una distribución Normal son:

```r
if (is.null(v_Norm)){
    cat("Ninguna")
  } else {
    v_Norm
  }
```

```
## Ninguna
```

Mientras que las variables que NO siguen una distribucion normal son estas otras

```r
if (is.null(v_noNorm)){
    cat("Ninguna")
  } else {
```

```
    v_noNorm
  }
```

```
## [1] "age"      "trtbps"   "chol"     "thalachh" "oldpeak"
```

A partir de estos datos Concluiremos que: Como en todas las variables de nuestro data set el p-value es menor que alfa (0.5), y siendo la hipotesis nula que la poblacion está distribuida normalmente, podemos decir que la hipotesis nula queda rechazada concluyendo que los datos no vienen de una distribucion normal.

**Tratamiento de outiliers** (pendiente del punto 3.2), al observar en los histogramas que las distribuciones son relativamente normales, considereamos adecuado normalizar las variables numericas, usando la estadarizacion **Z-SCORE**, esta estandarizacion transforma cada valor original en unidades de desviacion estandar con respecto a la media de la variable, los valores negativos indican que estan por debajo de la media de la variable original, con ello tambien conseguimos poner las variables en la misma escala.

```
##
#varNUM <- c("age", "trtbps", "chol", "thalachh", "oldpeak")
# Estandarización Z-score
heart_zscore <- heart
heart_zscore[, varNUM] <- scale(heart[, varNUM])
head(heart_zscore,5)
```

```
##          age sex cp      trtbps        chol fbs restecg   thalachh exng
## 1  0.9506240   1  3  0.76269408 -0.25591036   1       0 0.01541728    0
## 2 -1.9121497   1  2 -0.09258463  0.07208025   0       1 1.63077374    0
## 3 -1.4717230   0  1 -0.09258463 -0.81542377   0       0 0.97589950    0
## 4  0.1798773   1  1 -0.66277043 -0.19802967   0       1 1.23784920    0
## 5  0.2899839   0  0 -0.66277043  2.07861109   0       1 0.58297496    1
##      oldpeak slp caa thall output
## 1  1.0855423   0   0     1      1
## 2  2.1190672   0   0     2      1
## 3  0.3103986   2   0     2      1
## 4 -0.2063639   2   0     2      1
## 5 -0.3786180   2   0     2      1
```

```
#
##
```

Homogeneidad

Después de analizar la normalidad de los datos contenidos en las variables cuantitativas del conjunto de datos y se ha visto que ninguna de estas variables sigue una distribución Normal, para contrastar la homogeneidad de la varianza en poblaciones no Normales, los tests más recomendados son el de Leven (utilizando la mediana) o el test no paramétrico de Fligner-Killeen (también basado en la varianza).

A continuación, en este apartado, se contrasta la homogeneidad de la varianza mediante el test de Levene: Donde $H_0$: Varianzas iguales; $H_1$: Varianzas NO iguales

```
##
#Usamos la libreria car
library(car)
#
# Para un nivel de significancia (alpha) = 0.05
alpha <- 0.05
datos_numericos$output <- as.integer(heart$output)
# Aplicamos el test a cada una de las variables
leveneTest(y = datos_numericos$age, group = datos_numericos$output, center = "median")
```

```
## Levene's Test for Homogeneity of Variance (center = "median")
##        Df F value   Pr(>F)
## group   1  7.9854 0.005031 **
##       301
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
leveneTest(y = datos_numericos$trtbps, group = datos_numericos$output, center = "median")
```

```
## Levene's Test for Homogeneity of Variance (center = "median")
##        Df F value Pr(>F)
## group   1   1.857  0.174
##       301
```

```
leveneTest(y = datos_numericos$chol, group = datos_numericos$output, center = "median")
```

```
## Levene's Test for Homogeneity of Variance (center = "median")
##        Df F value Pr(>F)
## group   1  0.1015 0.7503
##       301
```

```
leveneTest(y = datos_numericos$thalachh, group = datos_numericos$output, center = "median")
```

```
## Levene's Test for Homogeneity of Variance (center = "median")
##        Df F value  Pr(>F)
## group   1  5.2467 0.02268 *
##       301
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
leveneTest(y = datos_numericos$oldpeak, group = datos_numericos$output, center = "median")
```

```
## Levene's Test for Homogeneity of Variance (center = "median")
##        Df F value    Pr(>F)
## group   1  32.916 2.344e-08 ***
##       301
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#
##
```

Por los resultados obtenidos podemos ver en este test las varianzas de las variables trtbps (p-value = 0.174), chol (p-value = 0.7503), thalachh (p-value =0.02268) no son significativamente diferentes entre si, ya que el valor de p-value es mayor a alfa (0.05), por tanto se cumple en estos casos el supuesto de homogeneidad de la varianza ($H_0$), mientras que para las variables age(p-value = 005031) y oldpeak (p-value= 0,002344) al ser su p-value menor que alfa se rechaza la $H_0$, concluyendo que las varianzas no son iguales por tanto no tienen homogeneidad.

## 4.3 Aplicación de pruebas estadísticas para comparar los grupos de datos.

**En función de los datos y el objetivo del estudio, aplicar pruebas de contraste de hipótesis, correlaciones, regresiones, etc. Aplicar al menos tres métodos de análisis diferentes.**

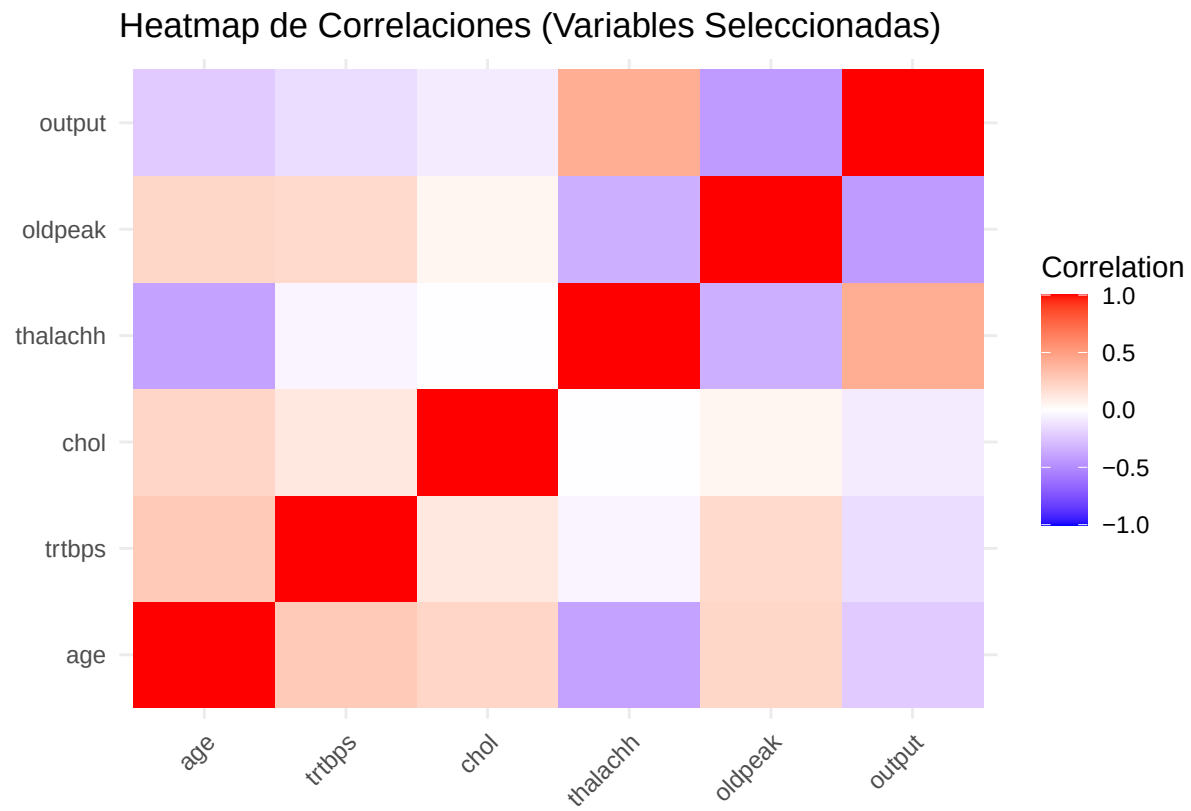
se aplicarán las pruebas estadísticas comentadas en el primer apartado del ejercicio 4 (Homogeneidad, Cálculo de la matriz de correlaciones entre las variables del conjunto de datos,..)

Hacemos un **HEATMAP**, para ver las correlaciones entre las variables continuas

```r
##
# Añadimos a nuestro dataFrame (datos_numericos) la variable OUTPUT.
datos_numericos$output <- as.integer(heart$output)

# Calculamos correlaciones
df_correl_seleccionadas <- as.data.frame(as.table(cor(datos_numericos)))

# Generamos Heatmap de correlaciones
ggplot(df_correl_seleccionadas, aes(x = Var1, y = Var2, fill = Freq)) +
  geom_tile() +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                       midpoint = 0, limit = c(-1, 1), space = "Lab",
                       name = "Correlation") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  ggtitle("Heatmap de Correlaciones (Variables Seleccionadas)")+
  ylab("") +
  xlab("")
```




```r
#
##
```

Observamos que no existen correlaciones fuertes entre las variables. La variable mas correlacionada con la variable objetivo **"output"** seria **"thalachh"**. Sin embargo, la variable **chol** apenas tiene correlacion con la varialbe objetivo, por lo que podemos plantearnos dercartarla por simplicidad del modelo.

Generacion de modelo de REGRESION LOGISTICA

Usamos un modelo de regresión logística para ver la relacion entre las variables factoriales con la variable objetivo ("output").

```
##
# Ajustamos el modelo de regresión logística
gml_1 <- glm(output ~ sex + fbs + exng + cp + restecg + slp + thall,
                      data = heart,
                      family = "binomial")
summary(gml_1)
```

```
##
## Call:
## glm(formula = output ~ sex + fbs + exng + cp + restecg + slp +
##     thall, family = "binomial", data = heart)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5160  -0.5313   0.2147   0.5589   2.4192
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.1423     0.9107   0.156 0.875803
## sex1         -1.2227     0.4256  -2.873 0.004067 **
## fbs1         -0.2215     0.4722  -0.469 0.639060
## exng1        -0.8435     0.3762  -2.242 0.024949 *
## cp1           1.4915     0.4946   3.016 0.002564 **
## cp2           1.8774     0.4232   4.436 9.14e-06 ***
## cp3           2.0776     0.6125   3.392 0.000694 ***
## restecg1      0.6696     0.3357   1.995 0.046084 *
## restecg2     -1.1033     1.5377  -0.717 0.473076
## slp1         -0.4522     0.6323  -0.715 0.474483
## slp2          0.9676     0.6572   1.472 0.140906
## thall2        0.4875     0.6460   0.755 0.450491
## thall3       -1.1646     0.6366  -1.829 0.067347 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 417.64  on 302  degrees of freedom
## Residual deviance: 240.13  on 290  degrees of freedom
## AIC: 266.13
##
## Number of Fisher Scoring iterations: 5
```

```
#
```

Tras los resultados obtenidos, comprobamos que las variables estadisticamente mas significativas son sex, exng, cp y restecg es decir son las que realmente tienen importancia a la hora de explicar la variabilidad de "output", esto lo vemos por los niveles de significancia representados por asteriscos (*** , ** , *, segun sean + significativos a - significativas), por esta razon vamos a descartar las variables fbs, slp, thall y chol

```
##
# Ajustamos el modelo de regresión logística
# Variables a incluir en el nuevo dataset
heart_clean_c <- c("age", "sex","cp",  "trtbps", "restecg",
```

```
                  "exng", "thalachh", "oldpeak", "output")

# Crear un nuevo dataset con las variables seleccionadas
heart_clean  <- heart_zscore[, heart_clean_c]

# Ver las primeras filas del nuevo dataset
head(heart_clean )
```

```
##          age sex cp      trtbps restecg exng    thalachh    oldpeak output
## 1  0.9506240   1  3  0.76269408       0    0  0.01541728  1.0855423      1
## 2 -1.9121497   1  2 -0.09258463       1    0  1.63077374  2.1190672      1
## 3 -1.4717230   0  1 -0.09258463       0    0  0.97589950  0.3103986      1
## 4  0.1798773   1  1 -0.66277043       1    0  1.23784920 -0.2063639      1
## 5  0.2899839   0  0 -0.66277043       1    1  0.58297496 -0.3786180      1
## 6  0.2899839   1  0  0.47760118       1    0 -0.07189928 -0.5508722      1
```

```
#
##
```

Modelo: RANDOM FOREST

Para finalizar el analisis, aplicaremos el modelo supervisado **RANDOM FOREST** para estimar el valor de output en nuevos datos. La elección de random forest se debe a que es robusto, puede ser mas rapido de entrenar y funciona bien con conjunto de datos pequeños.

Para aplicarlo,

1. Dividiremos los datos en un 80% datos de entrenamiento y en un 20% datos de prueba.
2. Dividimos los datos de entrenamiento en un 50% entrenamiento y 50% validacion, para garantizar que el modelo sea capaz de generalizar bien a datos nuevos y no vistos.
3. Finalmente, entrenamos el modelo.

```
##
# Creamos un índice de partición
set.seed(123)
particiones <- createDataPartition(heart_clean$output, p = 0.8, list = FALSE)

# Dividir el conjunto de datos en entrenamiento (80%) y prueba (20%)
datos_entrenamiento <- heart_clean[particiones, ]
datos_prueba <- heart_clean[-particiones, ]

# Dividir el conjunto de entrenamiento en entrenamiento y validación (50-50)
particiones_entrenamiento <- createDataPartition(datos_entrenamiento$output, p = 0.5, list = FALSE)
datos_entrenamiento_real <- datos_entrenamiento[particiones_entrenamiento, ]
datos_validacion <- datos_entrenamiento[-particiones_entrenamiento, ]

# Entrenar el modelo en el conjunto de entrenamiento real
modelo_rf <- randomForest(output ~ ., data = datos_entrenamiento_real)

#
##
```

Una vez creado el modelo, vamos a evaluarlo usando una matriz de confusion para analizar la precision, sensibilidad y especifidad. Usaremos los datos de prueba generados para dicha evaluacion.

```
##
# Se realizan predicciones en el conjunto de prueba
predicciones_prueba <- predict(modelo_rf, newdata = datos_prueba)
```

```r
# Comparar predicciones con etiquetas reales
matriz_confusion <- table(predicciones_prueba, datos_prueba$output)
print(matriz_confusion)
```

```
##
## predicciones_prueba  0  1
##                   0 18  2
##                   1  9 31
```

```r
# Calcular métricas de evaluación
precision <- (matriz_confusion[1,1] + matriz_confusion[2,2]) / sum(matriz_confusion)
sensibilidad <- matriz_confusion[2,2] / (matriz_confusion[2,1] + matriz_confusion[2,2])
especificidad <- matriz_confusion[1,1] / (matriz_confusion[1,1] + matriz_confusion[1,2])
#
##
```

Con este modelo se obtienen las siguiente metricas:
- Precision: 0.8166667
- Sensibilidad: 0.775
- Especificidad: 0.9

# 5 Representación de los resultados a partir de tablas y gráficas.

**Este apartado se puede responder a lo largo de la práctica, sin necesidad de concentrar todas las representaciones en este punto de la práctica**

Este apartado queda resuelto, con los diferentes graficos aportados e implementados en puntos anteriores, conforme se iban requiriendo, tal y como se indica en el resultado de la practica.

# 6 Resolución del problema

**A partir de los resultados obtenidos, ¿cuáles son las conclusiones? ¿Los resultados permiten responder al problema?**

A partir de la matriz de fonfusion obtenida en el punto 4, que arroja los datos siguientes:

Verdaderos Positivos (TP): 31 Verdaderos Negativos (TN): 18 Falsos Positivos (FP): 2 Falsos Negativos (FN): 9

obtenemos las metricas de precision, sensibilidad y especficidad que nos ayudaran a resolver este punto,

**La precisión** es la proporción de predicciones correctas entre todas las predicciones. Se calcula como (TP + TN) / TOTAL. En nuestro caso, es 81.67%. Esto significa que el 81.67% de las predicciones realizadas por el modelo fueron correctas.

**La sensibilidad** mide la proporción de verdaderos positivos respecto a todos los casos positivos reales. En este caso, es 0.775 o 77.5%. Esto significa que el modelo identificó correctamente al 77.5% de todos los casos positivos reales.

**La especificidad** mide la proporción de verdaderos negativos respecto a todos los casos negativos reales. Se calcula como TN / (TN + FP). En nuestro caso, es 0.9 o 90%. Esto significa que el modelo identificó correctamente al 90% de todos los casos negativos reales.

**En resumen**, el modelo tiene una precisión bastante buena (81.67%) y una especificidad alta (90%). Sin embargo, la sensibilidad es un poco más baja (77.5%), lo que indica que el modelo podría perder algunos casos positivos reales. Estos resultados proporcionan una visión general del rendimiento del modelo en el conjunto de prueba.

Probablemente se puedan mejorar los resultados ampliando la muestra ya que dispone tan solo de300 registros. Otras opciones seria aplicar otras tecnicas como XGboost o support vector machines y comparar con los resultados del random forest, aunque de nuevo, dado el tamaño de la muestra, intuimos que la mejora si es que se produce usando esos modelos, no sera tan significativa. Además se podría probar hacer featuring engineering para intentar mejorar la calidad del dataset o la validacion cruzada para obtener estimaciones mas robustas del rendimiento del modelo.

**Respondiendo** a la pregunta de si los resultados pueden responder el problema, podriamos decir que en general, teniendo en cuenta el tamaño de la muestra, el modelo funciona aceptablemente. Sin embargo, teniendo en cuenta que nos encontramos en un contexto de salud, cuyo objetivo es estimar si un usuario tiene un output con valor 1 ($>$ 50 % de estrechamiento del diámetro (menos posibilidades de sufrir un ataque cardíaco) o valor 0($<$ estrechamiento del diámetro (menos posibilidades de sufrir un ataque cardíaco), un error en la estimación puede conllevar consecuencias graves sobre el usuario. Un falso negativo, haría que una persona que necesite tratamiento, sea ignorado y un falso positivo, tendria como consecuencia medicar una persona sana.

Por tanto, para el contexto en el que nos encontramos, el modelo deberia tener una mayor presicion, sensibilidad y especifidad (al menos un 95% de media) para evitar consecuencias graves a cuasa de errores en la estimación.

# 7 Código

**Hay que adjuntar el código, preferiblemente en R, con el que se ha realizado la limpieza, análisis y representación de los datos. Si lo preferís, también podéis trabajar en Python**

El código de esta practica está incluido en este fichero con extension rmd, se puede localizar tambien en la siguiente URL:
https://github.com/pallaresl/PRA2_ASR_LPO/tree/main/code/PRA2_ASR_LPO.rmd

Los datos de salida se exportan mediante el siguiente comando (**write()**) y pueden ser descargados en GitHub desde la dirección: https://github.com/pallaresl/PRA2_ASR_LPO/tree/main/data/heart_clean.csv

```
write.csv(heart_zscore, file = "heart_clean.csv")
```

# 8 Vídeo

Realizar un breve vídeo explicativo de la práctica (máximo 10 minutos), donde ambos integrantes del equipo expliquen con sus propias palabras el desarrollo de la práctica, basándose en las preguntas del enunciado para justificar y explicar el código desarrollado. Este vídeo se deberá entregar a través de un enlace al Google Drive de la UOC (https://drive.google.com/drive/u/1/folders/1skOlQthXCBac6oybkI7KB8XbJt3RytIy),junto con enlace al repositorio Git entregado.

# 9 Contribuciones

Las contribuciones de los miembros del equipo en las tareas de la práctica son las siguientes:

| **Contribuciones** | **Firma** |
|---|---|
| Investigación previa | ASR, LPO |
| Redacción de las respuestas | ASR, LPO |
| Desarrollo código | ASR, LPO |
| Participacion en el video | ASR, LPO |