

# Tipología y ciclo de vida de los datos: Pr2 - Como realizar la limpieza y análisis de datos

Alejandro Hernández Slamerón, Almir Cáceres Barraquer

Junio 2023

- 1 Descripción del dataset
- 2 Integración y selección
- 3 Limpieza de datos
  - 3.1 Gestión de elementos erroneos
  - 3.2 Identificación y gestión de valores extremos
- 4 Análisis de los datos
  - 4.1 Selección de los datos
  - 4.2 Comprobación de la normalidad y homogeneidad de la varianza
  - 4.3 Aplicación de pruebas estadísticas
- 5 Representación de los resultados
- 6 Resolución del problema
- 7 Bibliografía

## PRÁCTICA 2 Introducción

### 1 Descripción del dataset

El juego de datos que se ha seleccionado está basado en los datos médicos de distintos pacientes referentes al **desarrollo de problemas cardíacos**. Estos datos presentan una serie de valores de los pacientes, basados en las **medidas de su estado físico**, así como de análisis de sangre efectuados. Entre otros se recogen datos numéricos como la medida de la frecuencia cardíaca, medida del colesterol, sexo, edad, y datos categóricos como el tipo de dolencia o el resultado del electrocardiograma. Con estos datos se obtiene un **valor del probabilidad de enfermedad cardíaca** en la que se puede encontrar un paciente.

El objetivo analítico de estos datos es obtener un **valor de el estado cardíaco** en el que se encuentra el paciente, y qué variables pueden afectar más a este tipo de dolencias. Con esto se podría intentar encontrar una relación del estado de un paciente para conseguir disminuir los riesgos que tiene un paciente de sufrir tales dolencias antes de que estas se produzcan o realizar un **seguimiento más exhaustivo** a los pacientes más críticos.

### 2 Integración y selección

Ahora vamos a ver los datos de que se dispone para realizar un primer estudio sobre las posibles decisiones previas que se han de tomar. Para ello cargamos el dataset y revisamos todos las variables que disponemos, para verificar cual podemos utilizar en estudios posteriores.

```
path = 'heart.csv'  
df_heart <- read.csv(path, row.names=NULL)
```

```
head(df_heart)
```

```

##  age sex cp trtbps chol fbs restecg thalachh exng oldpeak slpcaa 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
## 6 57 1 0 140 192 0 1 148 0 0.4 1 0 1 1

```

Vemos que disponemos de las siguientes variables para el estudio:

- **age** Age of the patient in years
- **sex** Sex of the patient
- **cp** chest pain type: (1=typical angina, 2=atypical angina, 3=non-anginal, 4=asymptomatic)
- **trtbps** resting blood pressure (in mm Hg)
- **chol** cholesterol in mg/dl fetched via BMI sensor
- **fbs** fasting blood sugar > 120 mg/dl (1 = true; 0 = false)
- **restecg** resting electrocardiographic results Value 0: normal, Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV), Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria
- **thalachh** maximum heart rate achieved
- **exan** exang: exercise induced angina (1 = yes; 0 = no)
- **oldpeak** ST depression induced by exercise relative to rest
- **slp** the slope of the peak exercise ST segment
- **caa** number of major vessels (0-3) colored by fluoroscopy
- **thall** 3normal; fixed defect; reversible defect
- **output** the predicted attribute (0 = less chance of heart attack, 1= more chance of heart attack)

Ahora se va a revisar que tipo de variable es cada una.

```
structure = str(df_heart)
```

```

## 'data.frame': 303 obs. of 14 variables:
## $ age      : int 63 37 41 56 57 57 56 44 52 57 ...
## $ sex      : int 1 1 0 1 0 1 0 1 1 1 ...
## $ cp       : int 3 2 1 1 0 0 1 1 2 2 ...
## $ 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      : int 1 0 0 0 0 0 0 1 0 ...
## $ restecg  : int 0 1 0 1 1 1 0 1 1 1 ...
## $ thalachh: int 150 187 172 178 163 148 153 173 162 174 ...
## $ exng    : int 0 0 0 0 1 0 0 0 0 0 ...
## $ oldpeak : num 2.3 3.5 1.4 0.8 0.6 0.4 1.3 0 0.5 1.6 ...
## $ slp     : int 0 0 2 2 2 1 1 2 2 2 ...
## $ caa     : int 0 0 0 0 0 0 0 0 0 0 ...
## $ thall   : int 1 2 2 2 2 1 2 3 3 2 ...
## $ output  : int 1 1 1 1 1 1 1 1 1 1 ...

```

Con esto tenemos la información acerca de la cantidad de datos de los que disponemos y su estructura. Como se puede observar, existen 14 variables y un total de 303 datos de estas variables. Además podemos concluir que todas las variables menos **oldpeak** son variables enteras, lo que nos es muy útil para la toma de decisión del tipo de estudio a realizar.

# 3 Limpieza de datos

Para poder iniciar el análisis de los datos, lo primero es revisar y arreglar los datos donde puedan faltar algún tipo de valores o estos sean erroneos (o puedan dar lugar a resultados erroneos)

## 3.1 Gestión de elementos erroneos

En primer lugar buscamos todos los datos que tengan alguna variable nulla o vacía.

```
print('Blancos')
```

```
## [1] "Blancos"
```

```
colSums(df_heart=="")
```

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

```
print('NA')
```

```
## [1] "NA"
```

```
colSums(is.na(df_heart))
```

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

Ahora vamos a revisar los valores de las variables para comprobar si alguna de ellos tuviese valores erroneos.

```
summary(df_heart)
```

```

##      age          sex         cp        trtbps
##  Min.   :29.00   Min.   :0.0000   Min.   :0.000   Min.   : 94.0
##  1st Qu.:47.50  1st Qu.:0.0000  1st Qu.:0.000  1st Qu.:120.0
##  Median :55.00  Median :1.0000  Median :1.000  Median :130.0
##  Mean   :54.37  Mean   :0.6832  Mean   :0.967  Mean   :131.6
##  3rd Qu.:61.00  3rd Qu.:1.0000  3rd Qu.:2.000  3rd Qu.:140.0
##  Max.   :77.00  Max.   :1.0000  Max.   :3.000  Max.   :200.0
##      chol         fbs       restecg      thalachh
##  Min.   :126.0  Min.   :0.0000  Min.   :0.0000  Min.   : 71.0
##  1st Qu.:211.0 1st Qu.:0.0000  1st Qu.:0.0000  1st Qu.:133.5
##  Median :240.0  Median :0.0000  Median :1.0000  Median :153.0
##  Mean   :246.3  Mean   :0.1485  Mean   :0.5281  Mean   :149.6
##  3rd Qu.:274.5 3rd Qu.:0.0000  3rd Qu.:1.0000  3rd Qu.:166.0
##  Max.   :564.0  Max.   :1.0000  Max.   :2.0000  Max.   :202.0
##      exng        oldpeak      slp        caa
##  Min.   :0.0000  Min.   :0.00  Min.   :0.000  Min.   :0.0000
##  1st Qu.:0.0000 1st Qu.:0.00  1st Qu.:1.000 1st Qu.:0.0000
##  Median :0.0000  Median :0.80  Median :1.000  Median :0.0000
##  Mean   :0.3267  Mean   :1.04  Mean   :1.399  Mean   :0.7294
##  3rd Qu.:1.0000 3rd Qu.:1.60  3rd Qu.:2.000  3rd Qu.:1.0000
##  Max.   :1.0000  Max.   :6.20  Max.   :2.000  Max.   :4.0000
##      thall        output
##  Min.   :0.000  Min.   :0.0000
##  1st Qu.:2.000  1st Qu.:0.0000
##  Median :2.000  Median :1.0000
##  Mean   :2.314  Mean   :0.5446
##  3rd Qu.:3.000  3rd Qu.:1.0000
##  Max.   :3.000  Max.   :1.0000

```

- **caa:** Esta variable nos dicen en la descripción del dataset que toma valores entre el 0 y el 3. Viendo que su valor máximo es 4, deberíamos eliminar los datos que tengan un valor 4 en esta variable y considerarlos erroneos.

Eliminamos los valores de caa referentes a 4

```
df_heart <- df_heart[!df_heart$caa == 4, ]
```

## 3.2 Identificación y gestión de valores extremos

Hacemos una última confirmación con el comando describe, para comprobar si hemos de revisar alguna variable más que pueda tener valores anómalos

```
describe(df_heart)
```

```
## df_heart
##
## 14 Variables      298 Observations
## -----
## age
##      n  missing distinct      Info      Mean      Gmd      .05      .10
##      298       0       41    0.999    54.51   10.29     40      42
##      .25       .50       .75      .90      .95
##      48       56       61       66      68
##
## lowest : 29 34 35 37 38, highest: 70 71 74 76 77
## -----
## sex
##      n  missing distinct      Info      Sum      Mean      Gmd
##      298       0       2    0.655    202    0.6779   0.4382
##
## -----
## cp
##      n  missing distinct      Info      Mean      Gmd
##      298       0       4    0.865    0.9597   1.105
##
## Value      0      1      2      3
## Frequency 142     49     84     23
## Proportion 0.477 0.164 0.282 0.077
## -----
## trtbps
##      n  missing distinct      Info      Mean      Gmd      .05      .10
##      298       0       49    0.994   131.6   19.48   108.0   110.0
##      .25       .50       .75      .90      .95
##      120.0    130.0   140.0   152.0   160.6
##
## lowest : 94 100 101 102 104, highest: 174 178 180 192 200
## -----
## chol
##      n  missing distinct      Info      Mean      Gmd      .05      .10
##      298       0      152       1   246.9   56.01   175.8   190.8
##      .25       .50       .75      .90      .95
##      211.0    241.5   275.0   309.0   327.4
##
## lowest : 126 131 141 149 157, highest: 394 407 409 417 564
## -----
## fbs
##      n  missing distinct      Info      Sum      Mean      Gmd
##      298       0       2    0.378     44    0.1477   0.2525
##
## -----
## restecg
##      n  missing distinct      Info      Mean      Gmd
##      298       0       3     0.76   0.5235   0.5281
##
## Value      0      1      2
## Frequency 146    148     4
```

```

## Proportion 0.490 0.497 0.013
## -----
## thalachh
##      n missing distinct      Info      Mean      Gmd      .05      .10
##    298      0      91        1    149.5    25.88   108.0   115.7
##    .25     .50     .75     .90     .95
##  133.0   152.5   165.8   177.3   182.0
##
## lowest : 71 88 90 95 96, highest: 190 192 194 195 202
## -----
## exng
##      n missing distinct      Info      Sum      Mean      Gmd
##    298      0      2     0.662      98  0.3289  0.4429
##
## -----
## oldpeak
##      n missing distinct      Info      Mean      Gmd      .05      .10
##    298      0      40     0.966    1.055    1.232      0.0      0.0
##    .25     .50     .75     .90     .95
##   0.0     0.8     1.6     2.8     3.4
##
## lowest : 0.0 0.1 0.2 0.3 0.4, highest: 4.0 4.2 4.4 5.6 6.2
## -----
## slp
##      n missing distinct      Info      Mean      Gmd
##    298      0      3     0.799    1.396  0.6309
##
## Value      0      1      2
## Frequency  21    138    139
## Proportion 0.070 0.463 0.466
## -----
## caa
##      n missing distinct      Info      Mean      Gmd
##    298      0      4     0.785    0.6745  0.9266
##
## Value      0      1      2      3
## Frequency  175    65    38    20
## Proportion 0.587 0.218 0.128 0.067
## -----
## thall
##      n missing distinct      Info      Mean      Gmd
##    298      0      4     0.779    2.312  0.6146
##
## Value      0      1      2      3
## Frequency  2     18    163    115
## Proportion 0.007 0.060 0.547 0.386
## -----
## output
##      n missing distinct      Info      Sum      Mean      Gmd
##    298      0      2     0.745      161  0.5403  0.4984

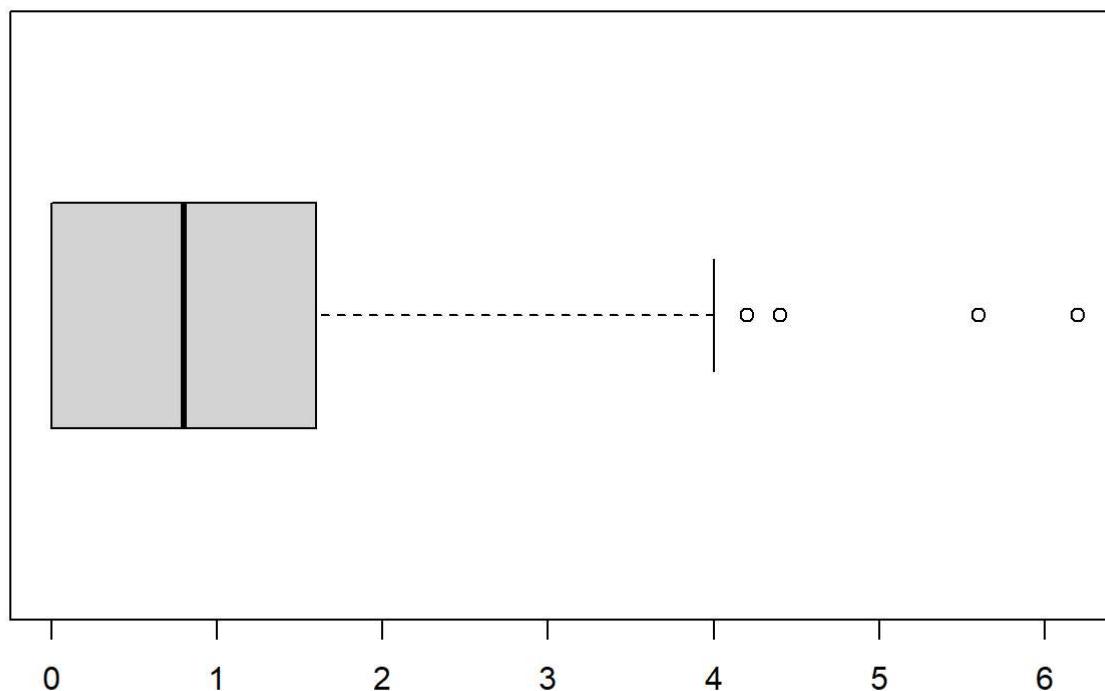
```

```
##  
## -----
```

- **oldpeak:** El valor máximo de esta variable es mucho más alto que la media. Esto no quiere decir que sea erróneo, pero es necesario revisar si estos valores altos son correctos.
- **chol:** Al igual que el anterior, el valor máximo de esta variable es mucho más alto que la media. Volvemos a hacer la misma suposición que para la variable anterior, ya que aunque sabemos que estos valores pueden ser altos en ciertas personas, no conocemos exactamente si estos valores son correctos y pueden afectar a los resultados del estudio.

Revisamos gráficamente la variable **oldpeak** para ver si tiene valores extremos.

```
boxplot(df_heart$oldpeak, horizontal = TRUE, outline=TRUE)
```

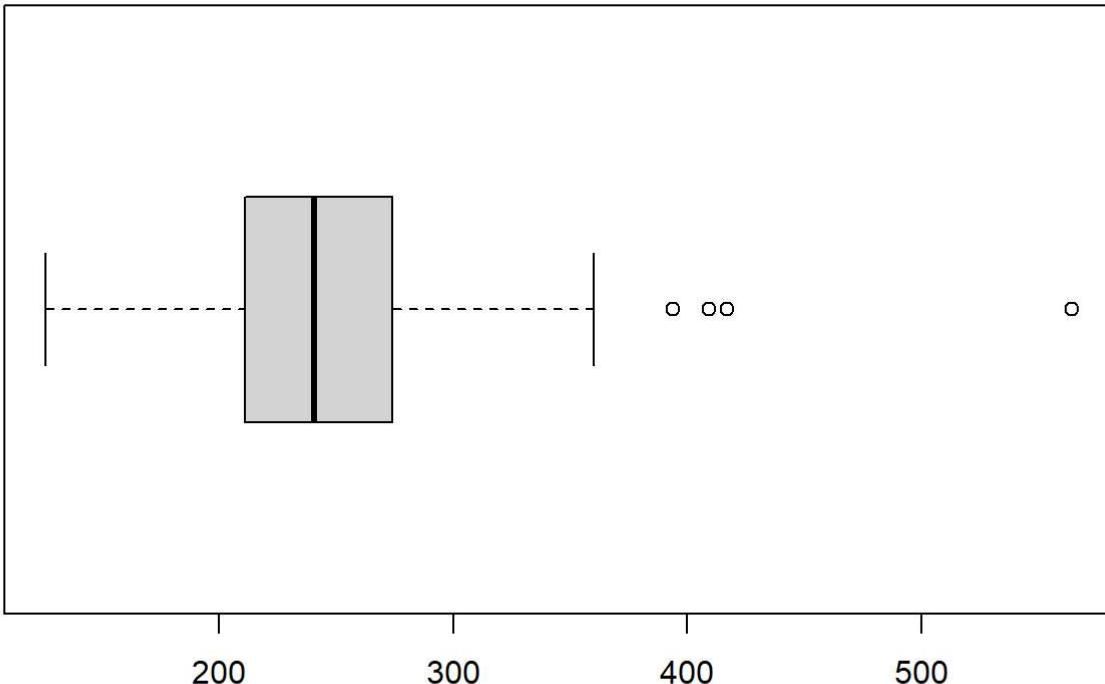


Podemos ver que existen 4 valores atípicos que tienen valores más altos de lo normal. Aunque desconocemos si teóricamente estos valores son o no correctos, podemos asumir que son valores anómalos y extraerlos del cálculo final, ya que no va a afectar considerablemente al resultado final.

```
df_heart <- df_heart[!df_heart$oldpeak >= 4, ]
```

Gráficamente también la variable **chol**

```
boxplot(df_heart$chol, horizontal = TRUE, outline=TRUE)
```



Al igual que con la variable **oldpeak**, estos valores no están significativamente separados. Aun así, se van a eliminar los valores anómalos.

```
df_heart <- df_heart[ !df_heart$chol >= 400, ]
```

## 4 Análisis de los datos

### 4.1 Selección de los datos

Una vez hecha la limpieza de datos, se pasa al proceso de análisis. En primer lugar se ha de definir cual es la finalidad que se busca a la hora del análisis. Para este caso, existe un valor de **output**, el cual toma valores de 0 y 1 y define menor o mayor probabilidad de tener enfermedades cardíacas respectivamente.

Para decidir el resto de variables que vamos a estudiar, vamos a revisar la distribución de las variables y la correlación que tienen entre ellas. Por ello y en lo referente a este estudio, se van a utilizar las variables cuantitativas, evitando escoger las variables con valores categóricos, las cuales se van a apartar para posibles estudios futuros más efectivos para este tipo de variables como los arboles de decisiones.

Por ello, las variables que se van a estudiar son las siguientes:

- **age** Podemos intuir que esta variable puede estar muy relacionada con las enfermedades cardíacas, ya que edades avanzadas se relacionan con aumentos de casos de este tipo
- **trtbps** La presión sanguínea tambien es una variable a tener en cuenta, puesto que altas presiones pueden derivar de/o problemas cardíacos.
- **chol** El colesterol es un factor también a tener en cuenta puesto que es un medidor muy utilizado para validar la salud de un paciente.

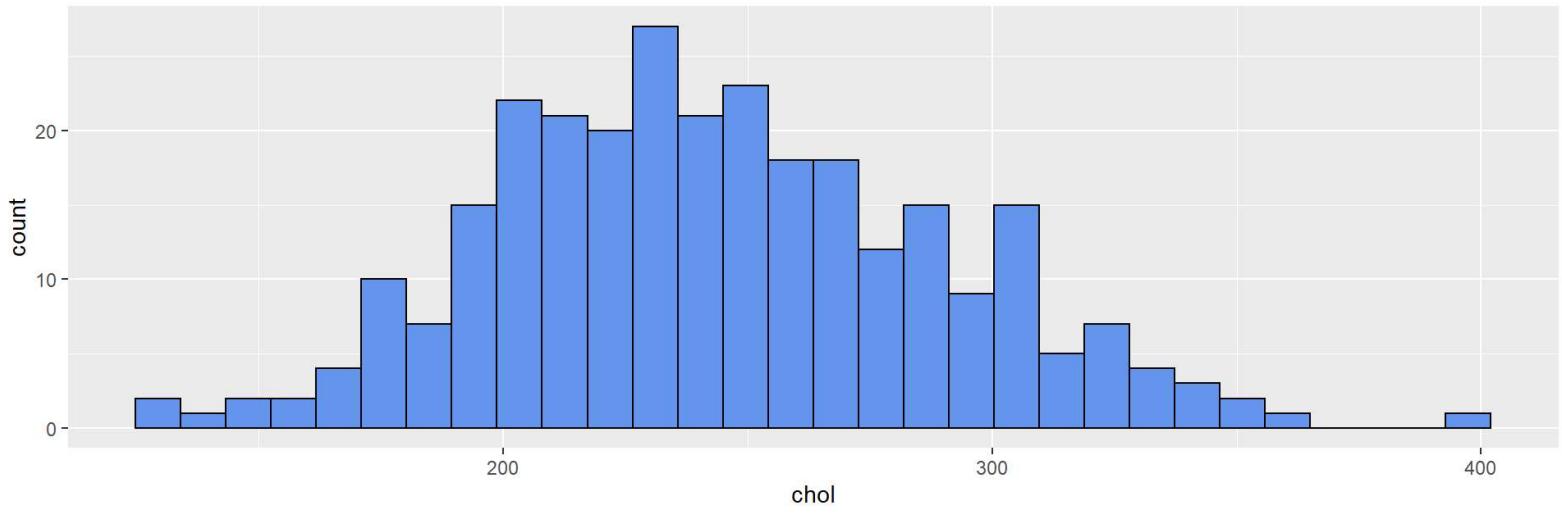
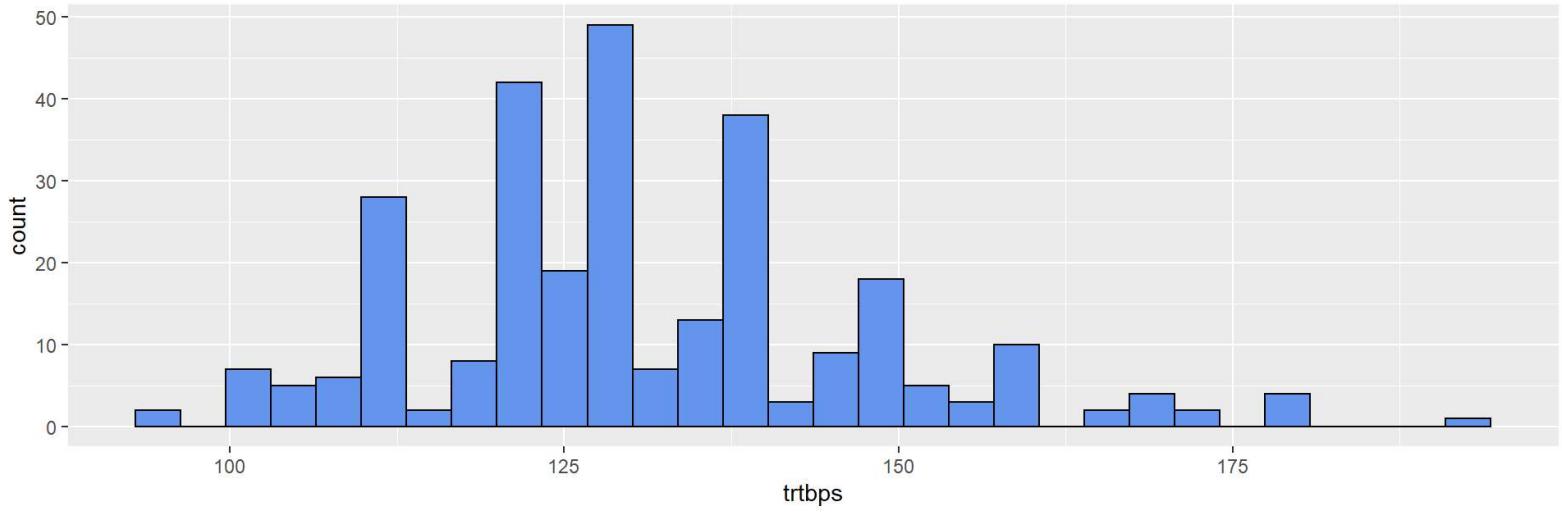
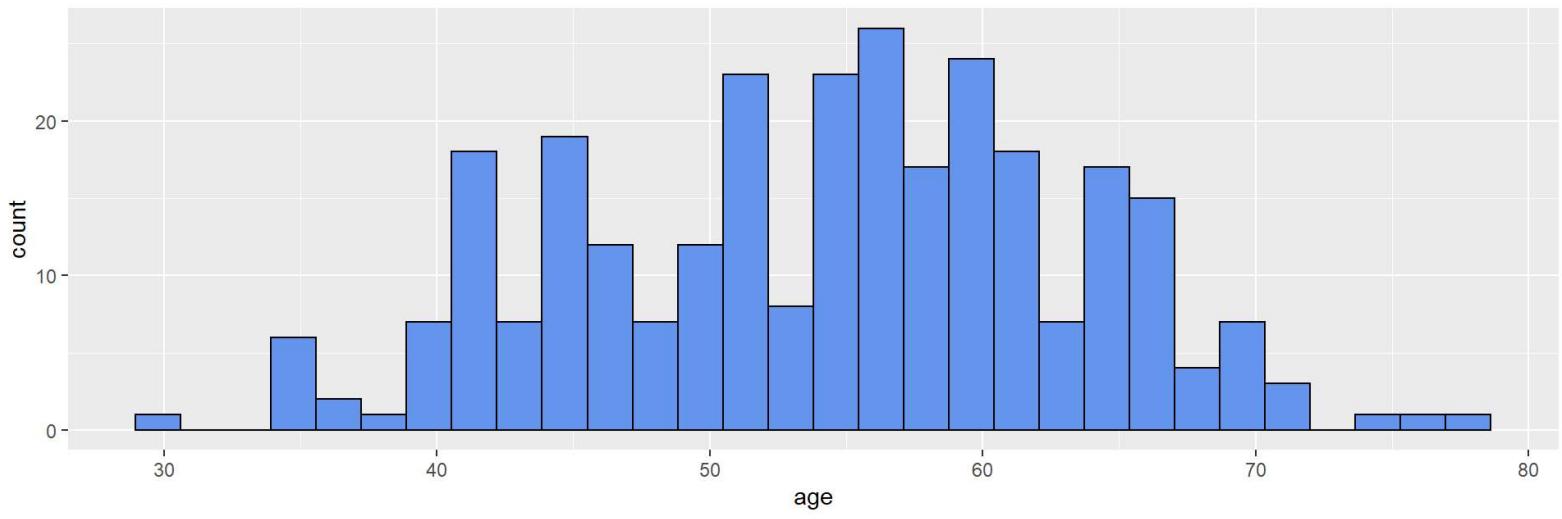
- **thalachh** No tenemos mucha información sobre esta variable, aún así podemos verificar si puede o no estar relacionada con este tipo de enfermedades.
- **oldpeak** Esta variable también la vamos a tener en cuenta puesto que sería un punto muy favorable tener un indicador directo de la medida de ritmo cardíaco para poder interpretar este tipo de dolencias cardíacas.
- **outpu** Este es el valor predicho que indica si existe riesgo o no sobre las dolencias cardíacas.

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

Ahora se va a comprobar la distribución de las variables, graficando estas para visualizar si siguen o no una distribución normal.

```
histList<- list()

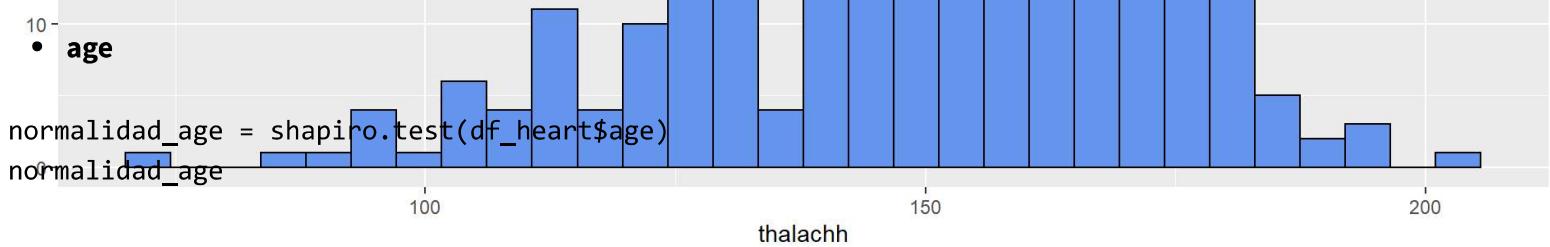
n = c("age","trtbps","chol","thalachh", "oldpeak")
df_heart_plot = df_heart %>% select(all_of(n))
for(y in 1:ncol(df_heart_plot)){
  col <- names(df_heart_plot)[y]
  ggp <- ggplot(df_heart_plot, aes_string(x = col)) +
    geom_histogram(bins = 30, fill = "cornflowerblue", color = "black",ggtitle = "Contador de ocurrencias por variable")
  histList[[y]] <- ggp # añadimos cada plot a la lista vacía
}
multiplot(plotlist = histList, coles = 1)
```



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

Viendo las gráficas podemos observar que la variable **oldpeak** no sigue una distribución normal. Para el resto necesitaríamos utilizar distintas herramientas para confirmar la asunción o no de normalidad de los datos.

También es posible aplicar pruebas de normalidad a las variables como la Shapiro-Wilk.



```
• age  
normalidad_age = shapiro.test(df_heart$age)  
normalidad_age
```

Un alto valor de estadístico W y un valor de p-value > 0,05 nos confirmarían que la distribución de esta variable es normal. En este caso no podemos afirmarlo, ya que aunque el estadístico W es alto, el valor de p-value es menor que el nivel de significancia, por lo que se rechaza la hipótesis nula de normalidad.

Esto no significa que no sea una distribución normal, ya que la prueba no es necesariamente concluyente.



```
normalidad_trtbps = shapiro.test(df_heart$trtbps)  
normalidad_trtbps
```

```
##  
## Shapiro-Wilk normality test  
##  
## data: df_heart$trtbps  
## W = 0.97109, p-value = 1.545e-05
```

Al igual que la variable anterior, no podemos afirmar que siga una distribución normal en base a la prueba.

- chol

```
normalidad_chol = shapiro.test(df_heart$chol)  
normalidad_chol
```

```
##  
## Shapiro-Wilk normality test  
##  
## data: df_heart$chol  
## W = 0.99311, p-value = 0.2099
```

En este caso tanto el estadístico W como el valor p-value están dentro del nivel de significancia para poder confirmar que se trata de una variable con distribución normal.

- **thalachh**

```
normalidad_thalachh = shapiro.test(df_heart$thalachh)
normalidad_thalachh
```

```
##
## Shapiro-Wilk normality test
##
## data: df_heart$thalachh
## W = 0.97485, p-value = 6.146e-05
```

En este caso, no se cumple el nivel de significancia del p-value, por lo que se desmiente la hipótesis de distribución normal.

- **oldpeak**

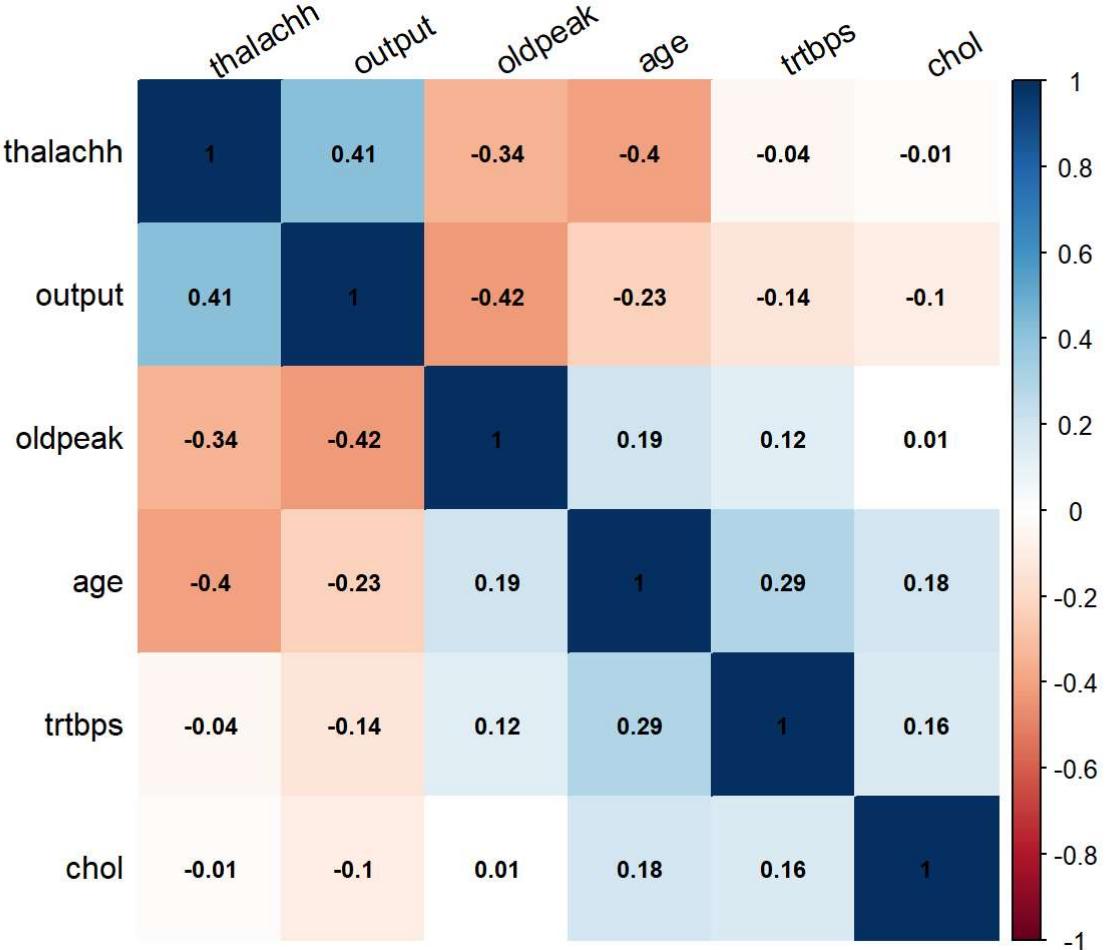
```
normalidad_oldpeak = shapiro.test(df_heart$oldpeak)
normalidad_oldpeak
```

```
##
## Shapiro-Wilk normality test
##
## data: df_heart$oldpeak
## W = 0.85934, p-value = 1.742e-15
```

Como ya habíamos supuesto al ver la representación gráfica, esta variable no sigue una distribución normal.

Ahora vamos a ver la correlación que tienen estos datos entre si. Una alta correlación entre variables para crear un modelo indicará que podemos eliminar una de ellas del estudio, ya que esto ayuda a simplificar el problema y evita el overfitting. Sin embargo, para realizar un estudio estadístico, una alta correlación puede ayudar a la precisión del estudio.

```
n = c("age", "trtbps", "chol", "thalachh", "oldpeak", "output")
factores= df_heart %>% select(all_of(n))
res<-cor(factores)
corrplot(res,method="color", tl.col="black", tl.srt=30, order = "AOE",
number.cex=0.75,sig.level = 0.01, addCoef.col = "black")
```



Vemos que ninguna de las variables esta altamente correlacionada, por lo que no podemos eliminar ninguna de estas variables para el estudio por medio de este criterio.

Aqui vemos que, para nuestra variable de predicción **output**, la variable con una mayor correlación es la variable **thalachh** seguida por la variable **age**, que si bien tiene una correlación negativa, es bastante significativa en términos absolutos.

## 4.3 Aplicación de pruebas estadísticas

Ahora se van a aplicar purebas estadísticas de contraste de hipótesis para la predicción de la probabilidad de tener una enfermedad cardíaca, en función del máximo valor de pulso cardíaco. Esto se va a realacionar con la edad. Las hipótesis son las siguientes:

- **Hipótesis nula:** Un valor alto de **thalachh** significa valor alto de edad.
- **Hipótesis alternativa:** Valor de **thalachh** no afecta al valor de la edad.

```
# Normalización
min_max_norm <- function(x) {
  (x - min(x)) / (max(x) - min(x))
}

# Aplicar La normalización
thalachh_norm <- min_max_norm(df_heart$thalachh)
age_norm <- min_max_norm(df_heart$age)
```

```

mean_thalachh <- mean(thalachh_norm)
sd_thalachh <- sd(thalachh_norm)

mean_age <- mean(age_norm)
sd_age <- sd(age_norm)

```

Media y desviación estandard

	thalachh	age
mean	0.601191584434928	0.527439024390244
sd	0.177144264177859	0.189625299303937

#Nivel de confianza

```

alpha <- 1-0.98
n_thalachh <- length(df_heart$thalachh)
n_output <- length(df_heart$output) # ALMIR: ¿ESTO NO SIRVE PARA NADA NO?

SE <- sd_thalachh / sqrt(n_thalachh)
zobs <- (mean_age - mean_thalachh)/SE
zobs

```

```
## [1] -7.053277
```

```

zcrit_L <- qnorm(alpha/2)
zcrit_U <- qnorm(1-alpha/2)
c(zcrit_L,zcrit_U)

```

```
## [1] -2.326348  2.326348
```

```

pvalue <- pnorm(abs(zobs), lower.tail = FALSE)*2
pvalue

```

```
## [1] 1.747522e-12
```

El valor de p-value para esta hipótesis es menor que el valor de significancia 0.05, por lo que en este caso se rechaza la hipótesis nula. Esto nos indica que, un alto valor en la edad no significa un alto valor de máximo ritmo cardíaco, cosa que está dentro de lo que se podría estipular.

Podemos usar también una regresión lineal para modelar la relación entre la variable dependiente (**output**) y las variables independientes. En este caso vamos a normalizar los datos para poder aplicar correctamente el modelo lm.

```
# Aplicar la normalización para todos las variables a estudiar para poder aplicar el modelo Lm
df_heart$age <- min_max_norm(df_heart$age)
df_heart$trtbps <- min_max_norm(df_heart$trtbps)
df_heart$chol <- min_max_norm(df_heart$chol)
df_heart$thalachh <- min_max_norm(df_heart$thalachh)
df_heart$oldpeak <- min_max_norm(df_heart$oldpeak)
```

Ahora se generan dos conjuntos, uno de test y otro de entrenamiento.

```
# Establecer una semilla para reproducir el valor aleatorio continuamente para poder verificar el modelo creado
set.seed(200)
```

```
# Definimos el tamaño de las muestras del juego de datos para el entrenamiento en el 80%
train_size <- round(0.8 * nrow(df_heart))
```

```
# Crear índices para el conjunto de entrenamiento y prueba
train_index <- sample(1:nrow(df_heart), train_size, replace = FALSE)
train_df <- df_heart[train_index, ]
test_df <- df_heart[-train_index, ]
dim(train_df)
```

```
## [1] 230 14
```

```
dim(test_df)
```

```
## [1] 57 14
```

```
lr_model1 <- lm( output ~ age + trtbps + chol + thalachh + oldpeak, data = train_df)
```

```
# Obtener el resumen del modelo
summary(lr_model1)
```

```

## 
## Call:
## lm(formula = output ~ age + trtbps + chol + thalachh + oldpeak,
##      data = train_df)
## 
## Residuals:
##    Min     1Q   Median     3Q    Max 
## -0.94278 -0.33990  0.08818  0.34158  0.95563 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  0.30856   0.18834   1.638   0.103    
## age         -0.07096   0.17594  -0.403   0.687    
## trtbps      -0.21049   0.18539  -1.135   0.257    
## chol        -0.18561   0.17550  -1.058   0.291    
## thalachh     0.91444   0.19129   4.780 3.17e-06 *** 
## oldpeak     -0.46787   0.11713  -3.994 8.80e-05 *** 
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
## 
## Residual standard error: 0.4335 on 224 degrees of freedom 
## Multiple R-squared:  0.2625, Adjusted R-squared:  0.246  
## F-statistic: 15.95 on 5 and 224 DF,  p-value: 1.955e-13

```

Como vemos en el modelo lm, no se ajusta nada al valor que se requiere, ya que hay variables que tiene un p-value > 0,05. Eliminando estas variables podremos ajustar mejor el valor del modelo.

```
lr_model2 <- lm( output ~ thalachh + oldpeak, data = train_df)
```

```
# Obtener el resumen del modelo
summary(lr_model2)
```

```

## 
## Call:
## lm(formula = output ~ thalachh + oldpeak, data = train_df)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -0.8952 -0.3472  0.1173  0.3160  0.9290
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  0.09423   0.12302   0.766   0.445    
## thalachh     0.95386   0.17536   5.439 1.38e-07 ***
## oldpeak     -0.49155   0.11579  -4.245 3.19e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4342 on 227 degrees of freedom
## Multiple R-squared:  0.25, Adjusted R-squared:  0.2434
## F-statistic: 37.83 on 2 and 227 DF, p-value: 6.606e-15

```

Si bien no en gran medida, el ajuste ha bajado un poco. Las variables no tenían mucho peso sin embargo si que aportaban positivamente. Vamos a evaluar ambos modelos.

```

predictions = predict(lr_model1, test_df)
tab = table(predictions>0.5, test_df$output)
tab

```

```

## 
##      0   1
## FALSE 15  8
## TRUE   9 25

```

```

acc = sum(diag(tab)) / sum(tab)
sprintf("Precisión: %s", acc)

```

```

## [1] "Precisión: 0.701754385964912"

```

```

sens = tab[2,2] / sum(tab[,2])
sprintf("Sensibilidad:%s", sens)

```

```

## [1] "Sensibilidad:0.757575757575758"

```

```

predictions = predict(lr_model2, test_df)
tab = table(predictions>0.5, test_df$output)
tab

```

```
##  
##          0   1  
##  FALSE 17   5  
##  TRUE   7  28
```

```
sum(diag(tab)) / sum(tab)
```

```
## [1] 0.7894737
```

```
acc = sum(diag(tab)) / sum(tab)  
sprintf("Precisión: %s", acc)
```

```
## [1] "Precisión: 0.789473684210526"
```

```
sens = tab[2,2] / sum(tab[,2])  
sprintf("Sensibilidad:%s", sens)
```

```
## [1] "Sensibilidad:0.848484848484849"
```

```
cat("")
```

Podemos ver que la eliminación de las variables ha mejorado el resultado, hemos pasado de 15 negativos y 25 positivos bien detectados a 17 y 28, respectivamente, ganando en total un 8% de precisión. El modelo es aceptable, la sensibilidad del modelo es del 84%, lo cual es bastante bueno ya que el factor crítico aquí es detectar los positivos, los falsos positivos no tienen tanta criticidad.

Por último se va a predecir todo el modelo para observar el resultado.

```
sum(tab[,2])
```

```
## [1] 33
```

```
predictions = predict(lr_model1, df_heart)  
tab = table(predictions>0.5, df_heart$output)  
tab
```

```
##  
##          0   1  
##  FALSE 82   33  
##  TRUE   47 125
```

```
acc = sum(diag(tab)) / sum(tab)  
sprintf("Precisión: %s", acc)
```

```
## [1] "Precisión: 0.721254355400697"
```

```
sens = tab[2,2] / sum(tab[,2])
sprintf("Sensibilidad:%s", sens)
```

```
## [1] "Sensibilidad:0.791139240506329"
```

```
predictions = predict(lr_model2, df_heart)
tab = table(predictions>0.5, df_heart$output)
tab
```

```
##
##          0    1
##  FALSE   81   27
##  TRUE    48 131
```

```
acc = sum(diag(tab)) / sum(tab)
sprintf("Precisión: %s", acc)
```

```
## [1] "Precisión: 0.738675958188153"
```

```
sens = tab[2,2] / sum(tab[,2])
sprintf("Sensibilidad:%s", sens)
```

```
## [1] "Sensibilidad:0.829113924050633"
```

Habiendo predicho todo el dataset podemos ver que la mejoría en el segundo modelo se reduce, aunque sigue estando presente. Al ser un modelo lineal vemos que además la presencia de sobrentrenamiento es nula, de hecho los resultados han empeorado al incluir todo el dataset.

La precisión en este caso es del 73% y la sensibilidad del 82%.

Ahora vamos a probar con un modelo lógico, entrenándolo con las variables categóricas.

```
logic_model = glm(formula=output~slp+thall+caa+exng+restecg+sex+cp+fbs, data=train_df, family=binomial)
summary(logic_model)
```

```

## 
## Call:
## glm(formula = output ~ slp + thall + caa + exng + restecg + sex +
##      cp + fbs, family = binomial, data = train_df)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.7192 -0.3915  0.1458  0.5152  2.7811
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2.2448    0.9918   2.263  0.023610 *
## slp          1.5179    0.3748   4.050 5.13e-05 ***
## thall       -1.0799    0.3290  -3.282  0.001030 **
## caa         -1.3283    0.2905  -4.573 4.82e-06 ***
## exng        -1.2898    0.4495  -2.869  0.004111 **
## restecg     -0.1510    0.3815  -0.396  0.692299
## sex         -1.8017    0.4828  -3.732  0.000190 ***
## cp            0.7843    0.2080   3.771  0.000163 ***
## fbs          0.2964    0.6106   0.486  0.627304
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 317.11 on 229 degrees of freedom
## Residual deviance: 162.78 on 221 degrees of freedom
## AIC: 180.78
##
## Number of Fisher Scoring iterations: 6

```

Vemos que el nivel de significancia de restecg y el de fbs son muy bajos, su valor p no está por debajo de 0.05. Vamos a representar también la criticidad de cada una de las variables ante la presencia de positivos.

```
exp(coefficients(logic_model))
```

	(Intercept)	slp	thall	caa	exng	restecg
##	9.4387359	4.5625639	0.3396159	0.2649255	0.2753215	0.8598811
##	sex	cp	fbs			
##	0.1650221	2.1909249	1.3450648			

Vemos que **slp**, **cp** y **fbs** son mayores que uno, lo que quiere decir que la influencia de estas variables es alta en la presencia de 1 como output.

```
logic_model2 = glm(formula=output~slp+thall+caa+exng+sex+cp, data=train_df, family=binomial)

summary(logic_model2)
```

```

## 
## Call:
## glm(formula = output ~ slp + thall + caa + exng + sex + cp, family = binomial,
##      data = train_df)
## 
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max 
## -2.7024 -0.4079  0.1409  0.5312  2.7448 
## 
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)    
## (Intercept)  2.2631    0.9454   2.394  0.016674 *  
## slp          1.4734    0.3655   4.031 5.55e-05 *** 
## thall        -1.0977   0.3255  -3.372  0.000745 *** 
## caa          -1.2939   0.2857  -4.528 5.95e-06 *** 
## exng         -1.2872   0.4487  -2.868  0.004125 **  
## sex          -1.7787   0.4804  -3.703  0.000213 *** 
## cp            0.7964    0.2051   3.883  0.000103 *** 
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
## Null deviance: 317.11  on 229  degrees of freedom 
## Residual deviance: 163.22  on 223  degrees of freedom 
## AIC: 177.22 
## 
## Number of Fisher Scoring iterations: 6

```

Eliminando las variables con bajo valor de significancia y repitiendo el modelo logístico, vemos que el valor de estadístico AIC disminuye, lo que significa una mayor calidad en el modelo. Vamos a comparar ahora la precisión de los dos modelos.

```

predictions = predict(logic_model, test_df)
tab = table(predictions>0.5, test_df$output)
tab

```

```

## 
##      0   1 
## FALSE 21  9 
## TRUE   3 24

```

```

acc = sum(diag(tab)) / sum(tab)
sprintf("Precisión: %s", acc)

```

```

## [1] "Precisión: 0.789473684210526"

```

```

sens = tab[2,2] / sum(tab[,2])
sprintf("Sensibilidad:%s", sens)

```

```
## [1] "Sensibilidad:0.727272727272727"
```

```
predictions = predict(logic_model2, test_df)
tab = table(predictions>0.5, test_df$output)
tab
```

```
##
##          0   1
## FALSE 21 10
## TRUE   3 23
```

```
acc = sum(diag(tab)) / sum(tab)
sprintf("Precisión: %s", acc)
```

```
## [1] "Precisión: 0.771929824561403"
```

```
sens = tab[2,2] / sum(tab[,2])
sprintf("Sensibilidad:%s", sens)
```

```
## [1] "Sensibilidad:0.696969696969697"
```

Vemos que los resultados son muy similares al modelo anterior, sin embargo la eliminación de variables que habíamos detectado que tenían un bajo valor de significancia en este caso empeoran sutilmente el resultado, repercutiendo con una pérdida de un casi 2% de precisión y 3% de sensibilidad.

Se presentan de nuevo los resultados para todo el dataset.

```
predictions = predict(logic_model, df_heart)
tab = table(predictions>0.5, df_heart$output)
tab
```

```
##
##          0   1
## FALSE 111 31
## TRUE   18 127
```

```
acc = sum(diag(tab)) / sum(tab)
sprintf("Precisión: %s", acc)
```

```
## [1] "Precisión: 0.829268292682927"
```

```
sens = tab[2,2] / sum(tab[,2])
sprintf("Sensibilidad:%s", sens)
```

```
## [1] "Sensibilidad:0.80379746835443"
```

```
predictions = predict(logic_model2, df_heart)
tab = table(predictions>0.5, df_heart$output)
tab
```

```
##
##          0   1
##  FALSE 111  32
##  TRUE   18 126
```

```
acc = sum(diag(tab)) / sum(tab)
sprintf("Precisión: %s", acc)
```

```
## [1] "Precisión: 0.825783972125436"
```

```
sens = tab[2,2] / sum(tab[,2])
sprintf("Sensibilidad:%s", sens)
```

```
## [1] "Sensibilidad:0.79746835443038"
```

El resultado para todo el dataset si que es mejor que del del modelo lineal y se dispersa un poco las pérdidas del modelo ajustado con menos variables. Vamos a añadir las dos variables discretas con las que hemos ajustado el modelo **lm** y observar si obtenemos una mejora en las predicciones.

```
logic_model3 = glm(formula=output~slp+thall+caa+exng+restecg+sex+cp+fbs+thalachh+oldpeak, data=train_df,
family=binomial)

predictions = predict(logic_model3, test_df)
tab = table(predictions>0.5, test_df$output)
tab
```

```
##
##          0   1
##  FALSE 19   8
##  TRUE   5 25
```

```
acc = sum(diag(tab)) / sum(tab)
sprintf("Precisión: %s", acc)
```

```
## [1] "Precisión: 0.771929824561403"
```

```
sens = tab[2,2] / sum(tab[,2])
sprintf("Sensibilidad:%s", sens)
```

```
## [1] "Sensibilidad:0.757575757575758"
```

```
predictions = predict(logic_model3, df_heart)
tab = table(predictions>0.5, df_heart$output)
tab
```

```
##
##          0    1
##  FALSE 111  27
##  TRUE   18 131
```

```
acc = sum(diag(tab)) / sum(tab)
sprintf("Precisión: %s", acc)
```

```
## [1] "Precisión: 0.843205574912892"
```

```
sens = tab[2,2] / sum(tab[,2])
sprintf("Sensibilidad:%s", sens)
```

```
## [1] "Sensibilidad:0.829113924050633"
```

Vemos que para el set de test no existe mejora, a pesar de que si la hay si utilizamos el conjunto entero, no podemos asegurar que el modelo sea mejor.

## 5 Representación de los resultados

Consideramos que la representación de los resultados ha sido realizada a lo largo de todo el documento, ya que se adapta mejor a una redacción natural de todos los apartados.

## 6 Resolución del problema

El objetivo de este trabajo es hacer un estudio del juego de datos **heart**, el cual tiene ciertas variables relacionadas con enfermedades cardíacas obtenidas de pacientes. Este juego de datos tiene una variable objetivo que indica si el paciente tiene riesgo de sufrir de este tipo de enfermedad.

Se ha hecho una limpieza del juego de datos para verificar que no hubiese datos erroneos, vacíos o nulos. También se han eliminado los datos anómalos que se han encontrado tanto en la variable **chol** como en **oldpeak**.

Tras esto se ha hecho una revisión de la distribución de los datos, donde se han graficado para comprobar la normalidad de estos. También se ha utilizado el estadístico Shapiro-Wilk el cual comprueba la normalidad de los datos. Se ha visto que, aunque la mayoría obtenían un p-valor mayor que el necesario para asumir la normalidad, todas excepto una tenían un valor del estadístico W muy alto, por lo que no podemos descartar la no normalidad de estas. Aun así, por el teorema del límite central podríamos asumir que las variables se comportan como distribución normal dada la gran cantidad de datos.

Tras esto se ha hecho una correlación de los datos numéricos para comprobar que variables pueden estar más relacionadas entre sí. Esto nos da la posibilidad escoger que variables son más sensibles al cambio de nuestra variable target, y de eliminar variables con mucha correlación que inducirían a un overfitting en los estudios realizados.

Se ha separado el dataset en dos partes, para tener grupo de entrenamiento y otro de test. Se ha realizado un modelo lineal de las variables numéricas, eliminando las variables poco representativas, obteniendo un modelo con precisión de casi el 80%. Aún así, con un valor de R<sup>2</sup>=0.25, se puede decir que el modelo lineal no se ajusta correctamente.

Se ha probado un modelo logístico obteniendo una precisión del 82%, con un valor del estadístico AIC=177. Esto nos dice que, aunque la precisión del modelo es alta, la calidad del modelo no lo es tanto en base al valor tan alto del estadístico.

Por último se han revisado los valores del los Odds-ratios para confirmar que variables pueden ser consideradas de protección o de riesgo.

Como conclusión final y tras la aplicación de varios estudios estadísticos, se ha llegado a la conclusión de que la variable target **output** no está del todo relacionada con las variables del dataset. Esto puede deberse a que, o bien no se han aplicado los modelos estadísticos correctos, o esta variable no se ha obtenido correctamente, lo que implica que la predicción que se pude hacer de ella sea muy baja.

## 7 Bibliografía

- Linear Models lm (<https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/lm>)
- glm Model (<https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/glm>)
- relevel (<https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/relevel>)
- sample (<https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/sample>)
- Split Data (<https://www.statology.org/train-test-split-r/>)
- ggplot2 (<https://rpubs.com/daniballari/ggplot>)