PRÁCTICA 2: Limpieza y análisis de datos

Autor: Ricardo Colin Pérez

Diciembre 2021

Índice

1. Descripción del dataset. ¿Por qué es importante y qué pregunta/problema pretende responder?

En esta segunda práctica trataremos un dataset con el fin de aprender a identificar los datos relevantes para un proyecto analítico y usar las herramientas de integración, limpieza, validación y análisis de las mismas.

Como sabemos, todo estudio analítico nace de una necesidad, y como hay pocas cosas tan importantes para el ser humano como la salud, nos centraremos en el estudio de datos de pacientes con el objetivo de diagnosticar enfermedades de forma temprana y más eficaz. Este análisis tiene un valor claro y no es otro que el de alargar la esperanza de vida de los pacientes. Para focalizar más el caso de estudio, nos centraremos en el análisis de datos ligados a un solo órgano: el corazón.

Por lo tanto, el primer paso será escoger el dataset que queremos tratar. En nuestro caso, nos hemos decantado por el siguiente dataset libre disponible en Kaggle: https://www.kaggle.com/lourenswalters/uci-heart-disease-data-set. El dataset que utilizaremos también se puede encontrar en el "UCI Machine Learning Repository" a través del siguiente enlace: https://archive.ics.uci.edu/ml/datasets/Heart+Disease. Está formado por 303 observaciones (filas o registros) de 14 variables (columnas) (13 de entrada y 1 de salida). A continuación se describe cada uno de estos 14 atributos:

- **Age** (numeric): edad de la persona.
- **Sex** (numeric): sexo de la persona.
- **Chest_Pain_Type** (numeric): tipo de dolor de pecho que experimenta la persona.
- **Resting_Blood_Pressure** (numeric): presión arterial en reposo (en mm Hg).
- **Serum_Cholesterol** (numeric): colesterol sérico (en mg/dl).
- **Fasting_Blood_Sugar** (numeric): nivel de azúcar en sangre en ayunas relativo a 120 mg/dl.
- **Resting_ECG** (numeric): resultados electrocardiográficos en reposo.
- **Max_Heart_Rate_Achieved** (numeric): frecuencia cardíaca máxima de la persona.
- **Exercise_Induced_Angina** (numeric): angina inducida por el ejercicio.
- **ST_Depression_Exercise** (numeric): depresión ST de la persona.
- **Peak_Exercise_ST_Segment** (numeric): la pendiente del segmento ST máximo del ejercicio.
- **Num_Major_Vessels_Flouro** (character): número de vasos mayores (0-3) coloreados por la fluroscopia.
- **Thalassemia** (character): forma de la talasemia.
- **Diagnosis_Heart_Disease** (integer): indica si el paciente padece o no una enfermedad cardíaca.

A partir de este dataset nos planteamos identificar cuando un paciente es propenso a sufrir de enfermedades del corazón. Por ello, se abordarán tres análisis diferentes:

1. Identificar cómo se relacionan las principales variables de los pacientes que están más ligadas a la salud del corazón.
2. Predecir si un paciente debe ser o no diagnosticado con una enfermedad cardíaca en base a algunas características de su historial médico.
3. Segmentar el conjunto de pacientes para formar grupos homogéneos a los que se les pueda aplicar un mismo diagnóstico.

2. Integración y selección de los datos de interés a analizar.

Una vez se ha introducido el dataset con el que vamos a trabajar, cargaremos los datos del archivo "processed.cleveland.data" obtenido del enlace de Kaggle "https://www.kaggle.com/lourenswalters/uci-heart-disease-data-set".

```
heartData <- read.csv("processed.cleveland.data", stringsAsFactors = FALSE,
                      strip.white = TRUE, header = FALSE)
```

Añadimos los nombres a las diferentes columnas del dataframe "heartData".

```
names(heartData) <- c("Age", "Sex", "Chest_Pain_Type", "Resting_Blood_Pressure",
                      "Serum_Cholesterol", "Fasting_Blood_Sugar", "Resting_ECG",
                      "Max_Heart_Rate_Achieved", "Exercise_Induced_Angina",
                      "ST_Depression_Exercise", "Peak_Exercise_ST_Segment",
                      "Num_Major_Vessels_Flouro", "Thalassemia",
                      "Diagnosis_Heart_Disease")
```

Mostramos la primera parte del dataframe "heartData".

```
head(heartData)
```

```
##   Age Sex Chest_Pain_Type Resting_Blood_Pressure Serum_Cholesterol
## 1  63   1               1                    145                233
## 2  67   1               4                    160                286
## 3  67   1               4                    120                229
## 4  37   1               3                    130                250
## 5  41   0               2                    130                204
## 6  56   1               2                    120                236
##   Fasting_Blood_Sugar Resting_ECG Max_Heart_Rate_Achieved
## 1                   1           2                     150
## 2                   0           2                     108
## 3                   0           2                     129
## 4                   0           0                     187
## 5                   0           2                     172
## 6                   0           0                     178
##   Exercise_Induced_Angina ST_Depression_Exercise Peak_Exercise_ST_Segment
## 1                       0                   2.3                        3
## 2                       1                   1.5                        2
## 3                       1                   2.6                        2
## 4                       0                   3.5                        3
## 5                       0                   1.4                        1
## 6                       0                   0.8                        1
##   Num_Major_Vessels_Flouro Thalassemia Diagnosis_Heart_Disease
## 1                      0.0         6.0                       0
## 2                      3.0         3.0                       2
## 3                      2.0         7.0                       1
## 4                      0.0         3.0                       0
## 5                      0.0         3.0                       0
## 6                      0.0         3.0                       0
```

Finalmente, mostramos el tipo de dato asignado a cada campo.

```
sapply(heartData, function(x) class(x))
```

```
##                      Age                      Sex          Chest_Pain_Type
##                "numeric"                "numeric"                "numeric"
##   Resting_Blood_Pressure        Serum_Cholesterol      Fasting_Blood_Sugar
##                "numeric"                "numeric"                "numeric"
##              Resting_ECG  Max_Heart_Rate_Achieved  Exercise_Induced_Angina
##                "numeric"                "numeric"                "numeric"
##   ST_Depression_Exercise Peak_Exercise_ST_Segment Num_Major_Vessels_Flouro
##                "numeric"                "numeric"              "character"
##              Thalassemia  Diagnosis_Heart_Disease
##              "character"                "integer"
```

Como se puede observar, el tipo de dato asignado a cada campo se corresponde con el descrito en la anterior sección.

En cuanto a la selección de datos de interés a analizar, creemos que todos los atributos presentes en el dataset se corresponden con características importantes a tener en cuenta a la hora de diagnosticar cualquier enfermedad del corazón. Es por ello que será conveniente tenerlos todos en consideración durante la realización del análisis.

---

3. Limpieza de los datos.

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

Una vez hemos cargado, visualizado, y explicado el dataset con el que vamos a trabajar, vamos a lidiar con uno de los problemas más habituales en el tratamiento previo de los datos: la ausencia de valores para un atributo determinado.

Realizando una simple inspección visual del dataset, hemos visto que existen valores desconocidos indicados con "?". Por ello mostraremos estadísticas de valores vacíos según el carácter "?".

```
colSums(heartData=="?")
```

```
##                      Age                      Sex          Chest_Pain_Type
##                        0                        0                        0
##   Resting_Blood_Pressure        Serum_Cholesterol      Fasting_Blood_Sugar
##                        0                        0                        0
##              Resting_ECG  Max_Heart_Rate_Achieved  Exercise_Induced_Angina
##                        0                        0                        0
##   ST_Depression_Exercise Peak_Exercise_ST_Segment Num_Major_Vessels_Flouro
##                        0                        0                        4
##              Thalassemia  Diagnosis_Heart_Disease
##                        2                        0
```

Vemos que los atributos "Num_Major_Vessels_Flouro" y "Thalassemia" tienen elementos vacíos. Trataremos la falta de datos simplemente eliminando aquellas filas que contengan algún valor desconocido. Tomamos la decisión de ignorar las observaciones con elementos vacíos porque estamos tratando con un dataset que tiene pocos registros con "?" (4 y 2 observaciones para "Num_Major_Vessels_Flouro" y "Thalassemia" respectivamente de un total de 303 observaciones).

No obstante, existen otras alternativas para hacer frente a la falta de información. Una de ellas sería reemplazar los valores que faltan en cada columna por el valor más frecuente de esa columna. Otra opción más radical sería ignorar una de estas dos variables con valores vacíos si se considera no significante o relevante.

Eliminamos todas las filas (observaciones) que contienen elementos vacíos. Es decir eliminamos las filas con "Num_Major_Vessels_Flouro" o "Thalassemia" como "?".

```
heartData_clean <- heartData[!(heartData$Num_Major_Vessels_Flouro == "?"
                               | heartData$Thalassemia == "?"),]
```

Volvemos a enumerar las filas del dataframe.

```
row.names(heartData_clean) <- 1:nrow(heartData_clean)
```

Finalmente convertimos el campo "Num_Major_Vessels_Flouro" de *character* a *numeric*.

```
heartData_clean$Num_Major_Vessels_Flouro <-
  as.numeric(heartData_clean$Num_Major_Vessels_Flouro)
```

Después de tratar la falta de datos, obtenemos un dataframe formado por 297 observaciones (6 observaciones menos que inicialmente).

3.2. Identificación y tratamiento de valores extremos.

Ahora nos centraremos en los valores extremos, es decir, aquellos datos que se encuentran muy alejados de la distribución normal de una variable o población.

Generalmente, se considera que cuando un valor se encuentra alejado 3 desviaciones estándar con respecto a la media del conjunto es un *outlier*. Por ello, representaremos los datos numéricos mediante gráficos de cajas (*boxplots*), con el objetivo de detectar dichos *outliers*.

Cargamos la librería "ggplot2" si ya la tenemos instalada.

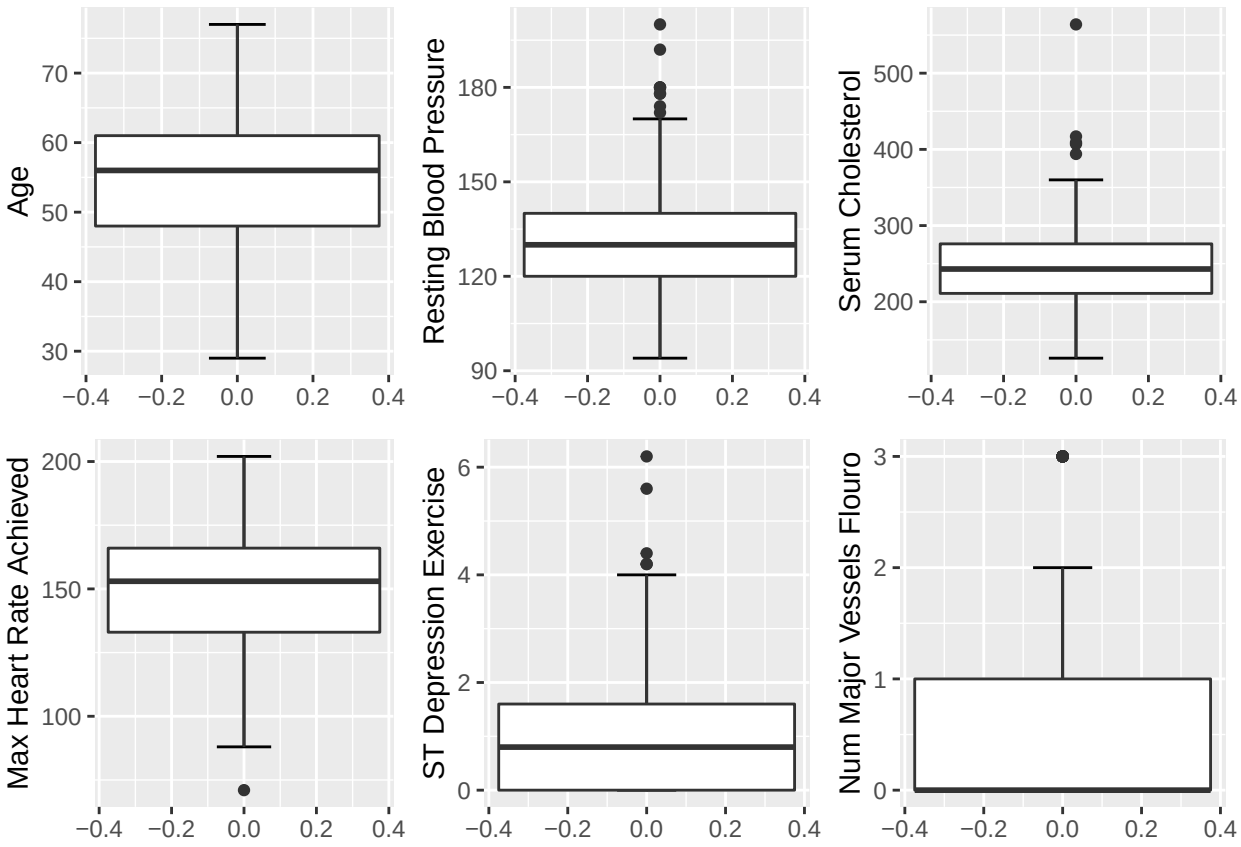
```
# https://cran.r-project.org/web/packages/ggplot2/index.html
if (!require('ggplot2')) install.packages('ggplot2'); library('ggplot2')
```

Seleccionamos las variables numéricas. Estas son "Age", "Resting_Blood_Pressure", "Serum_Cholesterol", "Max_Heart_Rate_Achieved", "ST_Depression_Exercise", y "Num_Major_Vessels_Flouro".

Utilizamos diagramas de cajas para representar los datos numéricos.

```
require(gridExtra)

Age <- ggplot(data = heartData_clean, aes(y = Age)) +
  stat_boxplot(geom = "errorbar", width = 0.15) +
  geom_boxplot() +
  ylab("Age")

Resting_Blood_Pressure <- ggplot(data = heartData_clean,
                                 aes(y = Resting_Blood_Pressure)) +
  stat_boxplot(geom = "errorbar", width = 0.15) +
  geom_boxplot() +
  ylab("Resting Blood Pressure")

Serum_Cholesterol <- ggplot(data = heartData_clean,
                            aes(y = Serum_Cholesterol)) +
  stat_boxplot(geom = "errorbar", width = 0.15) +
  geom_boxplot() +
  ylab("Serum Cholesterol")
```

```
Max_Heart_Rate_Achieved <- ggplot(data = heartData_clean,
                                  aes(y = Max_Heart_Rate_Achieved)) +
  stat_boxplot(geom = "errorbar", width = 0.15) +
  geom_boxplot() +
  ylab("Max Heart Rate Achieved")

ST_Depression_Exercise <- ggplot(data = heartData_clean,
                                 aes(y = ST_Depression_Exercise)) +
  stat_boxplot(geom = "errorbar", width = 0.15) +
  geom_boxplot() +
  ylab("ST Depression Exercise")

Num_Major_Vessels_Flouro <- ggplot(data = heartData_clean,
                                   aes(y = Num_Major_Vessels_Flouro)) +
  stat_boxplot(geom = "errorbar", width = 0.15) +
  geom_boxplot() +
  ylab("Num Major Vessels Flouro")

grid.arrange(Age, Resting_Blood_Pressure, Serum_Cholesterol,
             Max_Heart_Rate_Achieved, ST_Depression_Exercise,
             Num_Major_Vessels_Flouro, ncol=3)
```



Seguidamente vemos los números de estos valores extremos.

```
boxplot.stats(heartData_clean$Resting_Blood_Pressure)$out
```

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

```
boxplot.stats(heartData_clean$Serum_Cholesterol)$out
```

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

```
boxplot.stats(heartData_clean$Max_Heart_Rate_Achieved)$out
```

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

```
boxplot.stats(heartData_clean$ST_Depression_Exercise)$out
```

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

```
boxplot.stats(heartData_clean$Num_Major_Vessels_Flouro)$out
```

```
##  [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
```

Revisando los anteriores datos para varios pacientes, comprobamos que son valores que pueden darse. No obstante, apreciamos un valor de colesterol de una paciente por encima de los 500 mg/dl, lo que nos da a pensar que es un valor erróneo, ya que un colesterol total por encima de los 240 mg/dl se considera alto. Decidimos eliminar el registro de esta paciente para evitar resultados sesgados.

```
heartData_clean <- heartData_clean[!(heartData_clean$Serum_Cholesterol == 564),]
```

Volvemos a enumerar las filas del data frame.

```
row.names(heartData_clean) <- 1:nrow(heartData_clean)
```

Antes de pasar a analizar los datos, llevaremos a cabo un par de tareas más de preprocesado con el fin de trabajar con el dataset deseado.

- La primera de ellas será la de discretización: dado que cualquier valor superior a 0 en "Diagnosis_Heart_Disease" indica la presencia de una enfermedad cardíaca, agruparemos todos los valores mayores a 0, de modo que obtendremos un problema de clasificación binario (0 = ausencia de enfermedad cardíaca; 1 = presencia de enfermedad cardíaca).

Cargamos la librería "dplyr" si ya la tenemos instalada.

```
# https://cran.r-project.org/web/packages/dplyr/index.html
if (!require('dplyr')) install.packages('dplyr'); library('dplyr')
```

Creamos un campo que indique "Yes" o "No" haciendo referencia a los valores 1 o 0. Este nos servirá más adelante para hacer representaciones más visuales con la librería "ggplot2".

```
heartData_clean <- mutate(heartData_clean, Diagnosis_Heart_Disease_Present =
                            ifelse(Diagnosis_Heart_Disease >= 1, "Yes", "No"))
```

Agrupamos los valores superiores a 0 como 1.

```
heartData_clean <- mutate(heartData_clean, Diagnosis_Heart_Disease =
                            ifelse(Diagnosis_Heart_Disease >= 1, 1, 0))
```

Una vez que hemos realizado los procedimientos de integración, validación y limpieza sobre el dataset inicial, procedemos a guardar los datos en un nuevo fichero denominado "heartData_clean.csv".

```
write.csv(heartData_clean, "heartData_clean.csv")
```

- La segunda tarea de preprocesado que realizaremos será la de normalización de datos: consiste en situar los datos sobre una escala de valores equivalentes que permita la comparación de atributos que toman valores en dominios o rangos diferentes. Entre las diferentes técnicas existentes, nosotros llevaremos a cabo la normalización por la diferencia.

Aplicamos la normalización por la diferencia a las variables numéricas "Age", "Resting_Blood_Pressure", "Serum_Cholesterol", "Max_Heart_Rate_Achieved", "ST_Depression_Exercise" y "Num_Major_Vessels_Flouro". Estas variables serán escaladas entre el rango [0, 1].

```
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}

heartData_clean_norm <- heartData_clean

heartData_clean_norm$Age <- normalize(heartData_clean_norm$Age)
heartData_clean_norm$Resting_Blood_Pressure <-
  normalize(heartData_clean_norm$Resting_Blood_Pressure)
heartData_clean_norm$Serum_Cholesterol <-
  normalize(heartData_clean_norm$Serum_Cholesterol)
heartData_clean_norm$Max_Heart_Rate_Achieved <-
  normalize(heartData_clean_norm$Max_Heart_Rate_Achieved)
heartData_clean_norm$ST_Depression_Exercise <-
  normalize(heartData_clean_norm$ST_Depression_Exercise)
heartData_clean_norm$Num_Major_Vessels_Flouro <-
  normalize(heartData_clean_norm$Num_Major_Vessels_Flouro)
```

Guardamos los datos normalizados en un nuevo fichero denominado "heartData_clean_norm.csv".

```
write.csv(heartData_clean_norm, "heartData_clean_norm.csv")
```

---

4. Análisis de los datos.

4.1. Selección de los grupos de datos que se quieren analizar/comparar (planificación de los análisis a aplicar).

Ahora continuaremos con el análisis de los datos. A priori, utilizaremos todas las variables que disponemos para responder a los problemas planteados al principio. No obstante, seleccionaremos los grupos de datos dentro de nuestro dataset que pueden resultar interesantes para analizar y/o comparar.

Teniendo en cuenta la problemática identificada al inicio de la práctica, seleccionaremos dos grandes grupos en función de si un paciente debe ser o no diagnosticado con una enfermedad cardíaca.

```
heartData_disease <-
  heartData_clean[heartData_clean$Diagnosis_Heart_Disease_Present == "Yes",]

heartData_no_disease <-
  heartData_clean[heartData_clean$Diagnosis_Heart_Disease_Present == "No",]
```

También dividiremos las variables según sean categóricas o numéricas con el fin de aplicar diferentes métodos de análisis a cada uno de estos dos grupos con tipos de datos diferentes.

Empezamos seleccionando las variables categóricas. Estas son “Sex”, “Chest_Pain_Type”, “Fasting_Blood_Sugar”, “Resting_ECG”, “Exercise_Induced_Angina”, “Peak_Exercise_ST_Segment”, y “Thalassemia”.

```
heartData_categorical <- select(heartData_clean,
                                Sex,
                                Chest_Pain_Type,
                                Fasting_Blood_Sugar,
                                Resting_ECG,
                                Exercise_Induced_Angina,
                                Peak_Exercise_ST_Segment,
                                Thalassemia)
```

Seguidamente vamos a transformar los valores numéricos de las diferentes clases de cada uno de estos atributos a texto para que sean más fáciles de interpretar.

```
heartData_categorical <- mutate(heartData_categorical,
                                Sex = ifelse(Sex == 1, "Male", "Female"))

heartData_categorical$Chest_Pain_Type[
  heartData_categorical$Chest_Pain_Type == 1] <- "Typical"
heartData_categorical$Chest_Pain_Type[
  heartData_categorical$Chest_Pain_Type == 2] <- "Atypical"
heartData_categorical$Chest_Pain_Type[
  heartData_categorical$Chest_Pain_Type == 3] <- "Non-anginal"
heartData_categorical$Chest_Pain_Type[
  heartData_categorical$Chest_Pain_Type == 4] <- "Asymptomatic"

heartData_categorical <-
  mutate(heartData_categorical,
         Fasting_Blood_Sugar = ifelse(Fasting_Blood_Sugar
                                      == 1, "> 120 mg/dl", "<= 120 mg/dl"))

heartData_categorical$Resting_ECG[
  heartData_categorical$Resting_ECG == 0] <- "Normal"
heartData_categorical$Resting_ECG[
  heartData_categorical$Resting_ECG == 1] <- "ST-T abnormality"
heartData_categorical$Resting_ECG[
  heartData_categorical$Resting_ECG == 2] <- "LV hypertrophy"

heartData_categorical <-
  mutate(heartData_categorical,
         Exercise_Induced_Angina = ifelse(Exercise_Induced_Angina
                                          == 1, "Yes", "No"))

heartData_categorical$Peak_Exercise_ST_Segment[
  heartData_categorical$Peak_Exercise_ST_Segment == 1] <- "Upsloping"
heartData_categorical$Peak_Exercise_ST_Segment[
  heartData_categorical$Peak_Exercise_ST_Segment == 2] <- "Flat"
heartData_categorical$Peak_Exercise_ST_Segment[
  heartData_categorical$Peak_Exercise_ST_Segment == 3] <- "Downsloping"

heartData_categorical$Thalassemia[heartData_categorical$Thalassemia == 3] <- "Normal"
heartData_categorical$Thalassemia[heartData_categorical$Thalassemia == 6] <- "Fixed"
heartData_categorical$Thalassemia[heartData_categorical$Thalassemia == 7] <- "Reversable"
```

Finalmente, seleccionamos las variables numéricas. Estas son “Age”, “Resting_Blood_Pressure”, “Serum_Cholesterol”, “Max_Heart_Rate_Achieved”, “ST_Depression_Exercise”, y “Num_Major_Vessels_Flouro”.

```
heartData_numerical <- select(heartData_clean,
                              Age,
                              Resting_Blood_Pressure,
                              Serum_Cholesterol,
                              Max_Heart_Rate_Achieved,
                              ST_Depression_Exercise,
                              Num_Major_Vessels_Flouro)
```

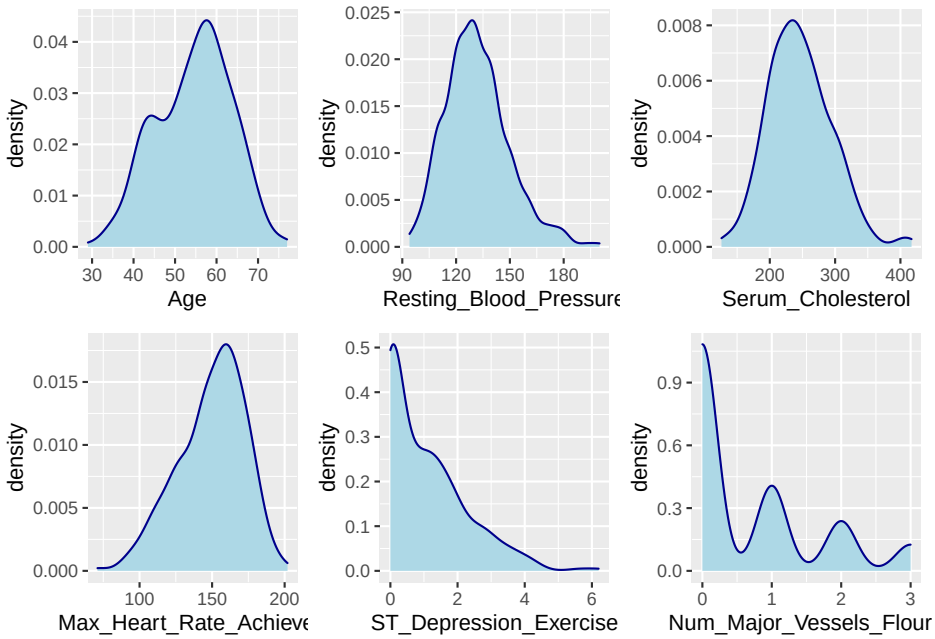
4.2. Comprobación de la normalidad y homogeneidad de la varianza.

La normalidad puede ser comprobada por inspección visual o mediante pruebas de significancia.

Como métodos visuales para comprobar la normalidad, podemos utilizar los diagramas de densidad y Q-Q (“quantile-quantile”).

El diagrama de densidad proporciona un juicio visual sobre si la distribución tiene forma de campana.

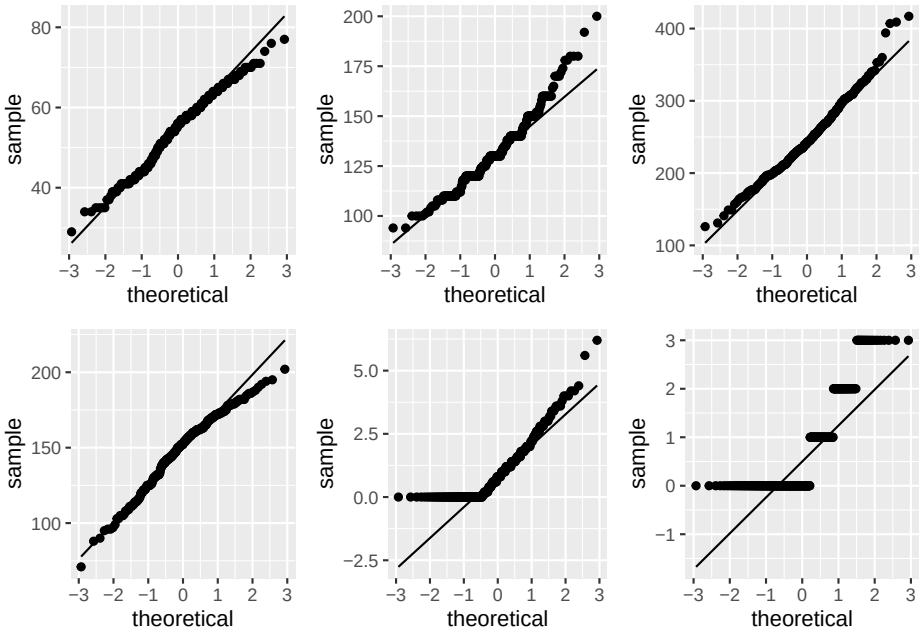
```
require(gridExtra)

Age <- ggplot(
  data = heartData_numerical, aes(x = Age)) +
  geom_density(color = "darkblue", fill = "lightblue")

Resting_Blood_Pressure <- ggplot(
  data = heartData_numerical, aes(x = Resting_Blood_Pressure)) +
  geom_density(color = "darkblue", fill = "lightblue")

Serum_Cholesterol <- ggplot(
  data = heartData_numerical, aes(x = Serum_Cholesterol)) +
  geom_density(color = "darkblue", fill = "lightblue")

Max_Heart_Rate_Achieved <- ggplot(
  data = heartData_numerical, aes(x = Max_Heart_Rate_Achieved)) +
  geom_density(color = "darkblue", fill = "lightblue")

ST_Depression_Exercise <- ggplot(
  data = heartData_numerical, aes(x = ST_Depression_Exercise)) +
  geom_density(color = "darkblue", fill = "lightblue")

Num_Major_Vessels_Flouro <- ggplot(
  data = heartData_numerical, aes(x = Num_Major_Vessels_Flouro)) +
  geom_density(color = "darkblue", fill = "lightblue")

grid.arrange(Age, Resting_Blood_Pressure, Serum_Cholesterol,
             Max_Heart_Rate_Achieved, ST_Depression_Exercise,
             Num_Major_Vessels_Flouro, ncol=3)
```



A simple vista, vemos que las cuatro primeras variables tienen una forma acampanada aunque no son del todo simétricas.

El diagrama Q-Q dibuja la correlación entre una muestra dada y la distribución normal. También se traza una línea de referencia de 45 grados.

```
require(gridExtra)

Age <- ggplot(
  data = heartData_numerical, aes(sample = Age)) + stat_qq() + stat_qq_line()

Resting_Blood_Pressure <- ggplot(
  data = heartData_numerical, aes(sample = Resting_Blood_Pressure)) +
  stat_qq() + stat_qq_line()

Serum_Cholesterol <- ggplot(
  data = heartData_numerical, aes(sample = Serum_Cholesterol)) +
  stat_qq() + stat_qq_line()

Max_Heart_Rate_Achieved <- ggplot(
  data = heartData_numerical, aes(sample = Max_Heart_Rate_Achieved)) +
  stat_qq() + stat_qq_line()

ST_Depression_Exercise <- ggplot(
  data = heartData_numerical, aes(sample = ST_Depression_Exercise)) +
  stat_qq() + stat_qq_line()

Num_Major_Vessels_Flouro <- ggplot(
  data = heartData_numerical, aes(sample = Num_Major_Vessels_Flouro)) +
  stat_qq() + stat_qq_line()

grid.arrange(Age, Resting_Blood_Pressure, Serum_Cholesterol,
             Max_Heart_Rate_Achieved, ST_Depression_Exercise,
             Num_Major_Vessels_Flouro, ncol=3)
```



Vemos que todos los puntos de las cuatro primeras variables caen aproximadamente a lo largo de esta línea de referencia, por lo que podríamos asumir la normalidad para estos cuatro atributos.

De todos modos, la inspección visual suele ser poco fiable. Por ello, también utilizamos pruebas de significancia que compare la distribución de la muestra con una normal para determinar si los datos muestran o no una desviación grave de la normalidad.

Existen varios métodos para la prueba de normalidad, como las pruebas de Kolmogorov-Smirnov y de Shapiro-Wilk.

El método de Shapiro-Wilk es ampliamente recomendado para la prueba de normalidad y proporciona una mejor potencia que la prueba de Kolmogorov-Smirnov. Se basa en la correlación entre los datos y las puntuaciones normales correspondientes.

Realizamos la prueba de normalidad de Shapiro-Wilk para cada una de las variables numéricas.

```
shapiro.test(heartData_numerical$Age)
```

```
shapiro.test(heartData_numerical$Resting_Blood_Pressure)
```

```
shapiro.test(heartData_numerical$Serum_Cholesterol)
```

```
shapiro.test(heartData_numerical$Max_Heart_Rate_Achieved)
```

```
shapiro.test(heartData_numerical$ST_Depression_Exercise)
```

```
shapiro.test(heartData_numerical$Num_Major_Vessels_Flouro)
```

A partir de los resultados, el valor $p < 0.05$ implica que la distribución de los datos es significativamente diferente de la distribución normal. En otras palabras, podemos asumir la no normalidad.

Para terminar, realizaremos la comprobación de la homocedasticidad, es decir, de la igualdad de varianzas entre los grupos que se van a comparar. Entre las pruebas más habituales se encuentran los tests de Leverne y de Fligner-Killeen. Nosotros aplicaremos el test de Fligner-Killeen, ya que se trata de la alternativa no paramétrica utilizada cuando los datos no cumplen con la condición de normalidad.

```
fligner.test(Age ~ Diagnosis_Heart_Disease_Present, data = heartData_clean)
```

```
fligner.test(Resting_Blood_Pressure ~ Diagnosis_Heart_Disease_Present,
             data = heartData_clean)
```

```
fligner.test(Serum_Cholesterol ~ Diagnosis_Heart_Disease_Present,
             data = heartData_clean)
```

```
fligner.test(Max_Heart_Rate_Achieved ~ Diagnosis_Heart_Disease_Present,
             data = heartData_clean)
```

```
fligner.test(ST_Depression_Exercise ~ Diagnosis_Heart_Disease_Present,
             data = heartData_clean)
```

```
fligner.test(Num_Major_Vessels_Flouro ~ Diagnosis_Heart_Disease_Present,
             data = heartData_clean)
```

Vemos que por un lado, las variables "Age", "Max_Heart_Rate_Achieved", "ST_Depression_Exercise" y "Num_Major_Vessels_Flouro" presentan varianzas estadísticamente diferentes para los diferentes grupos de "Diagnosis_Heart_Disease_Present" dado que las pruebas resultan en un p-valor inferior al nivel de significancia ($< 0{,}05$).

Por otro lado, las variables "Resting_Blood_Pressure" y "Serum_Cholesterol" presentan varianzas estadísticamente iguales para los diferentes grupos de "Diagnosis_Heart_Disease_Present" dado que las pruebas resultan en un p-valor mayor al nivel de significancia ($> 0{,}05$).

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.

A continuación, abordaremos los tres problemas diferentes que planteamos al inicio sobre el mismo dataset con el objetivo de diagnosticar enfermedades del corazón de forma temprana y más eficaz.

1. Análisis estadístico inferencial (correlación):

En este primer análisis investigaremos como se relacionan las variables numéricas entre ellas. Para ello, haremos uso de la correlación. Como ya hemos visto anteriormente, esta medida da una idea de las dependencias o asociaciones entre las variables continuas en forma de matriz. Cuantifica la asociación lineal entre un par de variables, es decir, la dependencia de una variable con respecto a la otra. Cuanto mayor sea la magnitud (más cercana a 1.00 o -1.00), más fuerte será la correlación. Un valor de 0 indica que no hay asociación entre las dos variables.

```
library(corrplot)
```

Aunque el coeficiente de correlación de Pearson es el más utilizado, requiere que la distribución de las variables sea normal. Teniendo en cuenta las pruebas de normalidad que hemos realizado anteriormente, nosotros calcularemos la correlación de Spearman, que aparece como una alternativa no paramétrica para medir el grado de dependencia entre dos variables.

```r
cor(heartData_numerical, method = "spearman")
```

```
##                                Age Resting_Blood_Pressure Serum_Cholesterol
## Age                      1.0000000             0.30761288        0.17649803
## Resting_Blood_Pressure   0.3076129             1.00000000        0.14687874
## Serum_Cholesterol        0.1764980             0.14687874        1.00000000
## Max_Heart_Rate_Achieved -0.3961909            -0.04557226       -0.03718179
## ST_Depression_Exercise   0.2491591             0.15948096        0.01951985
## Num_Major_Vessels_Flouro 0.3877971             0.07560454        0.14044054
##                          Max_Heart_Rate_Achieved ST_Depression_Exercise
## Age                                   -0.39619094             0.24915909
## Resting_Blood_Pressure                -0.04557226             0.15948096
## Serum_Cholesterol                     -0.03718179             0.01951985
## Max_Heart_Rate_Achieved                1.00000000            -0.43839893
## ST_Depression_Exercise                -0.43839893             1.00000000
## Num_Major_Vessels_Flouro              -0.28926055             0.26796468
##                          Num_Major_Vessels_Flouro
## Age                                    0.38779712
## Resting_Blood_Pressure                 0.07560454
## Serum_Cholesterol                      0.14044054
## Max_Heart_Rate_Achieved               -0.28926055
## ST_Depression_Exercise                 0.26796468
## Num_Major_Vessels_Flouro               1.00000000
```

Vemos que en ningún caso existen correlaciones lineales demasiado fuertes. Únicamente destacamos la relación directamente proporcional de la edad con presión arterial en reposo y el número de vasos mayores coloreados por la fluroscopia, así como la relación inversamente proporcional de la edad con la frecuencia cardíaca máxima.

2. Modelo supervisado (árbol de decisión):

En este segundo análisis, para reducir costes computacionales y trabajar en un entorno más justo, utilizaremos el dataset normalizado.

Primero eliminamos las variables que no nos interesan.

```r
heartData_clean_norm$Diagnosis_Heart_Disease_Present <- NULL
```

Aunque los datos estén desordenados, alteraremos la disposición de las filas para asegurarnos que habrá diversidad de ejemplos al dividir el dataset.

```r
set.seed(1)

heartData_random <- heartData_clean_norm[sample(nrow(heartData_clean_norm)),]
```

Seleccionamos las variables de entrada y la variable de salida "Diagnosis_Heart_Disease".

```r
set.seed(666)

y <- heartData_random[, "Diagnosis_Heart_Disease"]

X <- heartData_random[, c("Age", "Sex", "Chest_Pain_Type",
                          "Resting_Blood_Pressure", "Serum_Cholesterol",
                          "Fasting_Blood_Sugar", "Resting_ECG",
                          "Max_Heart_Rate_Achieved", "Exercise_Induced_Angina",
                          "ST_Depression_Exercise", "Peak_Exercise_ST_Segment",
                          "Num_Major_Vessels_Flouro", "Thalassemia")]
```

Dividimos los datos de entrada y de salida en una parte de "train" y otra de "test". La primera nos servirá para generar nuestro modelo a través de ejemplos, y la segunda para ponerlo a prueba con datos que nunca antes "ha visto", es decir, datos que no se han utilizado para su entreno.

```
split_prop <- 3
max_split <- floor(nrow(X)/split_prop)
tr_limit <- nrow(X)-max_split
ts_limit <- nrow(X)-max_split+1

trainX <- X[1:tr_limit,]
trainy <- y[1:tr_limit]
testX <- X[ts_limit:nrow(X),]
testy <- y[ts_limit:nrow(X)]
```

Comprobamos que la suma de filas de los subconjuntos de entrenamiento y de prueba sea igual al número de filas del conjunto original de datos.

```
dim(trainX)[1] + dim(testX)[1] == dim(X)[1]
```

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

Creamos un modelo de clasificación C5.0 que nos servirá para obtener un conjunto de reglas.

```
trainy = as.factor(trainy)

model <- C50::C5.0(trainX, trainy)

summary(model)
```

Por un lado, "Errors" muestra el número y porcentaje de casos mal clasificados en el subconjunto de entrenamiento. El árbol obtenido clasifica erróneamente 14 de los 198 casos dados, una tasa de error del 7.1 %.

3. Modelo no supervisado (*k-means*):

Para realizar el último análisis usaremos un modelo no supervisado. Es decir, un modelo que se genera sin conocimiento a priori. Entre los diferentes métodos de agrupamiento, emplearemos *k-means* para intentar segmentar el conjunto de pacientes para formar grupos homogéneos a los que se les pueda aplicar un mismo diagnóstico. *K-means* es un algoritmo de clasificación no supervisada (clusterización) que agrupa objetos en k grupos basándose en sus características. El agrupamiento se realiza minimizando la suma de distancias entre cada objeto y el centroide de su grupo o clúster.

En este caso también nos interesará trabajar con los datos normalizados. Es decir, sobre una escala de valores equivalentes que permita la comparación de atributos que toman valores en dominios o rangos diferentes. Seleccionamos los atributos numéricos.

```
heartData_num_norm <- heartData_clean_norm

heartData_num_norm$Sex <- NULL
heartData_num_norm$Chest_Pain_Type <- NULL
heartData_num_norm$Fasting_Blood_Sugar <- NULL
heartData_num_norm$Resting_ECG <- NULL
heartData_num_norm$Exercise_Induced_Angina <- NULL
heartData_num_norm$Peak_Exercise_ST_Segment <- NULL
heartData_num_norm$Thalassemia <- NULL
heartData_num_norm$Diagnosis_Heart_Disease <- NULL
```

Lo primero que hacemos es cargar la librería *cluster* que contiene las funciones que se necesitan.

```
library(cluster)
```

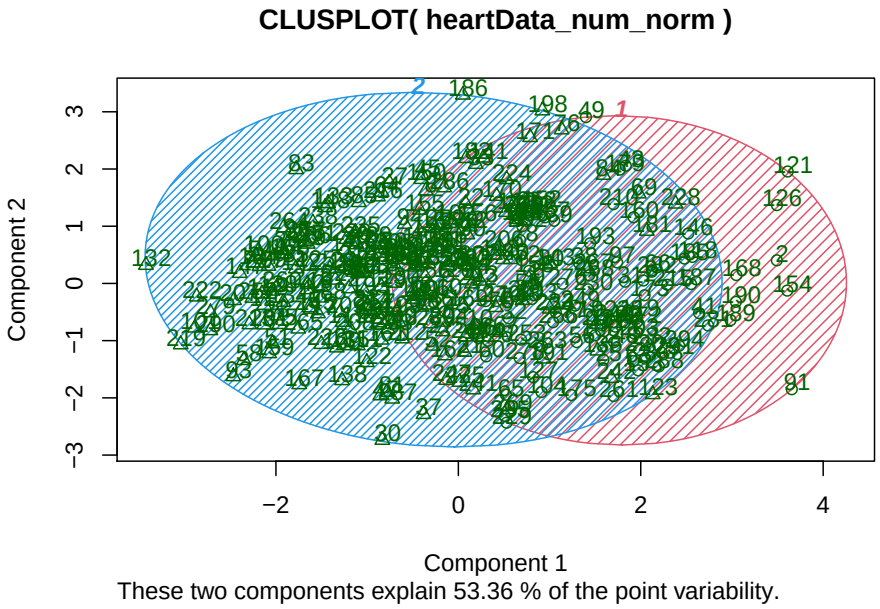
Ahora nos centraremos en el problema de distinguir dos grupos de pacientes según si deben ser diagnosticados con una enfermedad cardíaca o no. En este caso, el valor de "k" sería 2, perteneciendo cada uno de estos clústeres a las 2 clases que queremos agrupar.

```
d <- daisy(heartData_num_norm)
fit           <- kmeans(heartData_num_norm, 2)
y_cluster     <- fit$cluster
sk            <- silhouette(y_cluster, d)
resultado <- mean(sk[,3])
```

Después de llevar a cabo el análisis distinguiremos dos grupos de pacientes en función de sus condiciones físicas. Por ello, seguidamente estudiaremos los resultados encontrados con los 2 clústeres.

Visualizamos la agrupación con 2 clústeres mediante la función *clusplot*.

```
set.seed(123)
cl2_euclidean <- kmeans(heartData_num_norm, 2)
clusplot(heartData_num_norm, cl2_euclidean$cluster, color = TRUE, shade = TRUE, labels = 2, lines = 0)
```



La función *clusplot* utiliza análisis de componentes principales (PCA) para dibujar los datos. Por ello utiliza los dos primeros componentes principales para explicar los datos. Estos son los ejes ortogonales que a lo largo de ellos los datos tienen la mayor variabilidad (si los datos fueran en 2D, dos componentes principales podrían explicar el 100 % de la variabilidad de los datos).

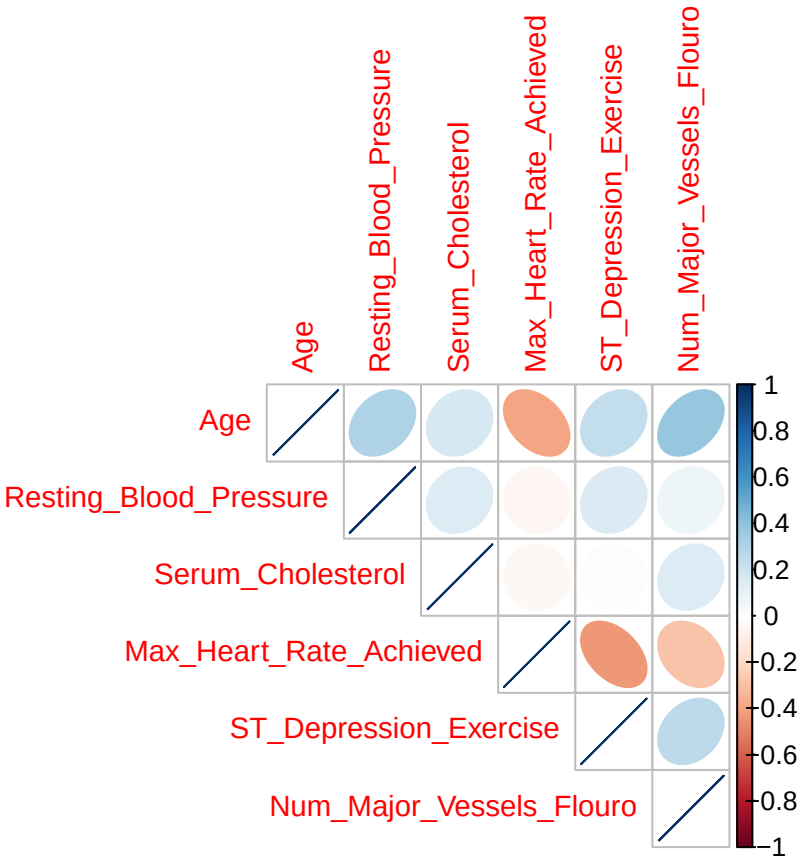
Podemos ver que los 2 clústeres se solapan entre ellos. Es en estas zonas de solape donde nuestro modelo de agregación puede confundir pacientes de diferentes segmentos. Por consiguiente, estaremos interesados en producir clústeres con bajas distancias intraclúster (alta similitud intraclúster) y altas distancias interclúster (baja similitud interclúster).

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

1. Análisis estadístico inferencial (correlación):

En esta sección representaremos los resultados obtenidos durante el análisis de una forma más visual. Para empezar, mostraremos la matriz de correlación que hemos obtenido antes a través de la función *corrplot*, la cual nos generará una gráfica mucho más fácil de comprender.

```
corrplot(cor(heartData_numerical, method = "spearman"),
         type="upper", method="ellipse", tl.cex=0.9)
```

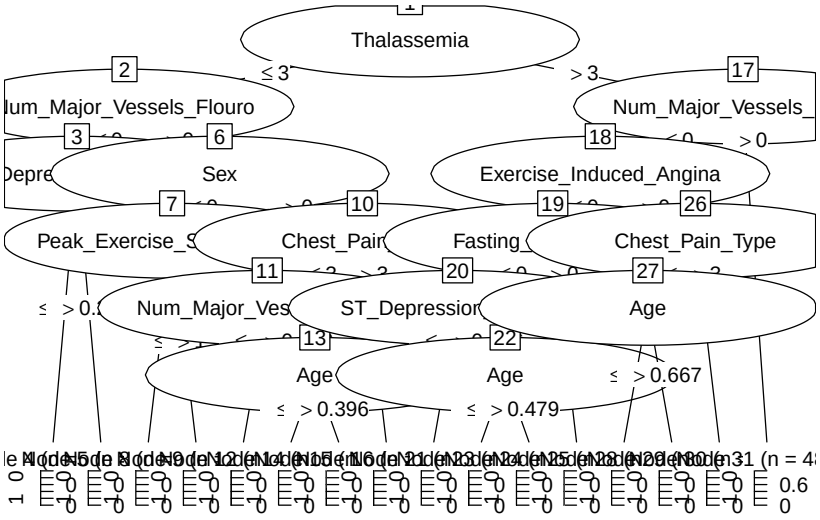


A simple vista observamos el comportamiento de cada uno de los pares de atributos. Vemos cuáles tienen una mayor dependencia y cuál es su pendiente según si son directa (por ejemplo, la relación entre "Age" y "Resting Blood Pressure") o inversamente proporcionales (por ejemplo, la relación entre "Age" y "Max Heart Rate Achieved").

2. Modelo supervisado (árbol de decisión):

En cuanto a los resultados del segundo análisis, primero mostraremos el árbol de decisión obtenido.

```
model <- C50::C5.0(trainX, trainy, noGlobalPruning = TRUE)

plot(model)
```



Como resultado se obtiene un árbol con un número de nodos tan grande que es difícil distinguir cada una de sus ramas y hojas.

Una vez tenemos el modelo, podemos comprobar su calidad prediciendo la clase para los datos de prueba que nos hemos reservado al principio.

```
predicted_model <- predict(model, testX, type="class")

print(sprintf("La precisión del árbol es: %.4f %%",
              100*sum(predicted_model == testy) / length(predicted_model)))
```

```
## [1] "La precisión del árbol es: 76.5306 %"
```

Vemos que la precisión obtenida para el primer modelo es de 76,5 % con nuevos datos. A continuación analizaremos la calidad de la predicción mediante una matriz de confusión que identifica los tipos de errores cometidos.

```
mat_conf <- table(testy, Predicted=predicted_model)

mat_conf
```

```
##      Predicted
## testy  0  1
##     0 38 13
##     1 10 37
```

De la matriz de confusión obtenemos:

- 38 verdaderos positivos (pacientes clasificados como sanos cuando están sanos)
- 37 verdaderos negativos (pacientes clasificados como enfermos cuando están enfermos)
- 13 falsos positivos/error tipo I (pacientes clasificados como sanos cuando están enfermos)
- 10 falsos negativos/error tipo II (pacientes clasificados como enfermos cuando están sanos)

3. Modelo no supervisado (*k-means*):

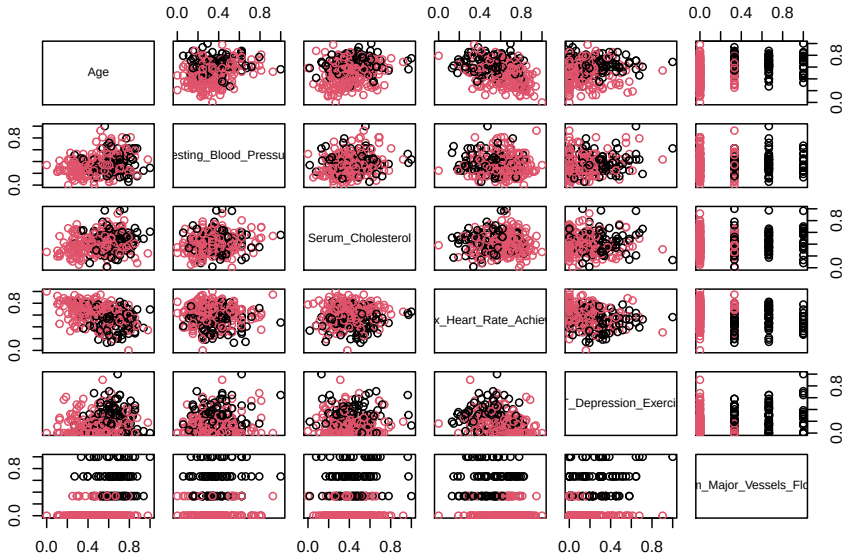
Finalmente, mostramos información sobre el resultado de la agrupación.

```
print(cl2_euclidean)
```

La salida impresa muestra las medias o centros de los clústeres en una matriz, cuyas filas son el número de clúster (de 1 a 2) y las columnas son las variables que hemos seleccionado. También podemos ver el *clustering vector*, que es un vector de enteros (de 1 a 2) que indica el clúster al que se asigna cada punto.

Ahora vamos a mostrar los clústeres encontrados entre pares de características. Esta representación nos permitirá ver que variables son buenos indicadores para diferenciar los diferentes segmentos.

```
heartData_num_norm$cluster <- NULL
with(heartData_num_norm, pairs(heartData_num_norm, col=c(1:2)[cl2_euclidean$cluster]))
```



En base al problema que estamos abordando, considerando que se trata de una segmentación de pacientes de una clínica u hospital, 2 clústeres son suficientes para clasificar los diferentes perfiles de personas. Para finalizar, podemos describir los 2 grupos a partir de la última gráfica.

Grupo 1 (rojo): pacientes sanos

- Suele ser gente joven
- Generalmente con baja presión arterial en reposo
- Bajos niveles de colesterol sérico
- Alta frecuencia cardíaca máxima
- Bajos niveles de depresión ST
- Sin vasos mayores coloreados por la fluroscopia

Grupo 2 (negro): pacientes propensos a sufrir enfermedades cardíacas

- Suele ser gente adulta
- Generalmente con alta presión arterial en reposo
- Altos niveles de colesterol sérico
- Baja frecuencia cardíaca máxima
- Altos niveles de depresión ST
- Con vasos mayores coloreados por la fluroscopia

6. Resolución del problema. A partir de los resultados obtenidos, ¿cuáles son las conclusiones? ¿Los resultados permiten responder al problema?

En esta práctica nos hemos centrado en el estudio de datos de pacientes con el objetivo de diagnosticar enfermedades del corazón. Hemos visto que la edad es un importante factor de riesgo en desarrollar enfermedades cardiovasculares. Más ligados con el corazón, hemos podido ver de las reglas obtenidas del árbol de decisión como otras variables como el dolor de pecho o la presión arterial en reposo son decisivas a la hora de hacer el diagnóstico del paciente. La primera porque cuando al corazón no le llega sangre lo suficientemente oxigenada, produce una especie de presión en el pecho, y la segunda debido a que la alta presión puede dañar las arterias.

A su vez, mediante un análisis estadístico inferencial de correlación, hemos sido capaces de ver cómo se relacionan las principales variables de los paciente que están más ligadas a la salud del corazón. También, hemos diseñado un modelo de clasificación para predecir si un paciente debe ser o no diagnosticado con una enfermedad cardíaca en base a algunas características de su historial médico. Finalmente, hemos segmentado un conjunto de pacientes para formar grupos homogéneos a los que se les pueda aplicar un mismo diagnóstico.

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.

Carpeta con el código generado para analizar los datos: https://github.com/RicardoCoPe/Pr-ctica-2-Limpieza-y-an-lisis-de-datos.

Contribuciones	Firma
Investigación previa	RCP
Redacción de las respuestas	RCP
Desarrollo del código	RCP