# Trabajo Práctico Integrador.

El trabajo que aquí se presenta integra toda la materia cursada en la asignatura Introducción a la Ciencia de Datos. Los datasets que se emplearán para estos menesteres son:

- **Heart**: Para las cuestiones relativas a **Clasificación**
- **Delta_ail:** Para los apartados relacionados con **Regresión**.

Por otra parte, para realizar las tareas relacionadas con la parte de **Análisis de datos**, se utilizarán ambos datasets. A continuación, se documentan las tareas realizadas con estos dos conjuntos de datos correspondientes a cada una de las partes que integran la asignatura.

## 1. Análisis de Datos

**Realizar las siguientes tareas para los datasets Heart y Delta_ail:**

**A-. Descripción del tipo de datos de entrada (lista, data frame, ect., numero de filas, columnas, tipo de datos atómicos, etc.)**

En primer lugar, se ha procedido a cargar los ficheros que contienen los datos de ambos datasets en las variables Heart y Delta. Para ello, se ha ejecutado el siguiente código R:

```
setwd("~/Dropbox/Master Ciencia de Datos/Introducción a la Ciencia de
Datos/Trabajos -100 por cien de la nota-/Trab Integrador T-P -95 por
ciento-/LopezGomezJulioAlbertoTP95")
Heart <- read.csv("heart.dat", comment.char = "@")
head(Heart)
Delta <- read.csv("delta_ail.dat", comment.char = "@")
head(Delta)
```

Para ello, tal y como se puede ver en el fragmento de código anterior, se ha modificado el directorio de trabajo por el directorio en el que se encuentran los ficheros con los datasets y se han guardado estos en las variables Heart y Delta, mostrando una pequeña visualización de los dos conjuntos de datos, que aglutina las seis primeras filas de cada dataset, operación que se realiza mediante la función head().

Una vez que los datasets que van a ser utilizados a lo largo de este trabajo han sido cargados, se analizarán los datos que contiene cada uno de ellos. En primer lugar, utilizando la función class() será posible comprobar a qué tipo de datos R pertenecen los datasets cargados. Esta operación viene dada por el siguiente fragmento de código.



*Julio Alberto López Gómez*
*Juan Carlos Cubero, Coral del Val, Rocío Romero, Rafael Alcalá*

```
> class(Heart)
[1] "data.frame"
> class(Delta)
[1] "data.frame
```

A continuación, mediante la función dim(), se obtendrá la dimensión en filas y columnas de cada uno de los datasets, lo cual permitirá tener una idea de las dimensiones de los datos con los que se está trabajando. Esta operación viene dada por el siguiente fragmento de código

```
> dim(Heart)
[1] 269  14
> dim(Delta)
[1] 7128    6
```

Donde el resultado de estas operaciones es un vector de dos componentes: la primera de ellas representa la dimensión del dataset en filas y la segunda la dimensión en columnas. Para este caso concreto, se puede observar que el dataset Heart es un dataset más corto que Delta ya que tiene muchas menos filas, pero se trata de un dataset más ancho que Delta, ya que éste solo tiene seis columnas o atributos, mientras que Heart posee catorce atributos.

A la luz de estos pequeños cálculos, ya es posible plantear una idea global sobre estos conjuntos de datos. Hasta ahora, se tienen dos dataframes, uno de ellos con pocas filas o registros pero un número de atributos considerable, mientras que el segundo se trata de un dataset más largo pero también más estrecho. Estas condiciones intrínsecas a los propios datos, serán importantes a la hora de realizar las tareas de clasificación y regresión de los siguientes epígrafes.

Por otra parte, si obtenemos una visualización de los datasets mediante la función head(), se puede comprobar que el nombre de las columnas que hacen referencia a los atributos aparecen codificados. Por este motivo, y para mejorar la legibilidad de las visualizaciones posteriores, se va a proceder a cambiar el nombre de las columnas de ambos datasets por otros más representativos, que han sido extraídos de los ficheros de datos de cada dataset. Estas operaciones son realizadas tal y como muestra el siguiente fragmento de código.

```
colnames(Heart) <- c("Age","Sex", "ChestPainType",
"RestBloodPressure", "SerumCholestoral",
"FastingBloodSugar","ResElectrocardio","MaxHeartRate","ExcerciseInduce
d","OldPeak","Slope","MajorVessels","Thal","Class")
head(Heart)

colnames(Delta) <- c("RollRate","PitchRate", "CurrPitch", "CurrRoll",
"DiffRollRate", "Sa")
head(Delta)
```

Una vez cambiados los nombres de las columnas, si utilizamos la función head(), ejecutando el fragmento de código que se adjunta a continuación, se obtendrá la salida que se muestra en las imágenes siguientes.

```
> head(heart)
> head(Delta)
```

Siendo las salidas de estos comandos, respectivamente:

```
> head(Heart)
  Age Sex ChestPainType RestBloodPressure SerumCholestoral FastingBloodSugar ResElectrocardio MaxHeartRate ExcerciseInduced OldPeak Slope MajorVessels Thal
1  67   0            3              115             564               0               2         160               0      16     2            0    7
2  57   1            2              124             261               0               0         141               0       3     1            0    7
3  64   1            4              128             263               0               0         105               1       2     2            1    7
4  74   0            2              120             269               0               2         121               1       2     1            1    3
5  65   1            4              120             177               0               0         140               0       4     1            0    7
6  56   1            3              130             256               1               2         142               1       6     2            1    6
  Class
1     1
2     2
3     1
4     1
5     1
6     2
```

```
> head(Delta)
  RollRate PitchRate CurrPitch CurrRoll DiffRollRate     Sa
1  -0.0023     0.0003      0.022   -0.011      0.00005  3e-04
2   0.0059    -0.0006      0.014   -0.005      0.00017 -2e-04
3   0.0046     0.0001      0.011    0.015     -0.00009 -2e-04
4  -0.0076     0.0012      0.010    0.001     -0.00010  2e-04
5  -0.0073     0.0013      0.010   -0.013      0.00006  1e-04
6   0.0054     0.0027      0.017   -0.022      0.00014 -1e-04
```

Por último, y no por ello menos importante, se ha pretendido dar una estructura de cada uno de los datasets, incluyendo el número y el nombre de cada una de las variables o atributos que contiene así como su tipo. Para ello, se ha utilizado la función str(). Además, para futuras conclusiones y cálculos, se ha creado una matriz que contiene los rangos de valores de las diferentes variables o atributos de cada dataset. Esto se ha conseguido mediante la función apply(), aplicando la función range() sobre cada uno de los atributos de cada dataset. El resultado de estas operaciones se muestra en el siguiente fragmento de código.

```
> str(Heart)
'data.frame':     269 obs. of  14 variables:
 $ Age              : int  67 57 64 74 65 56 59 60 63 59 ...
 $ Sex              : int  0 1 1 0 1 1 1 1 0 1 ...
 $ ChestPainType    : int  3 2 4 2 4 3 4 4 4 4 ...
 $ RestBloodPressure: int  115 124 128 120 120 130 110 140 150 135 ...
 $ SerumCholestoral : int  564 261 263 269 177 256 239 293 407 234 ...
 $ FastingBloodSugar: int  0 0 0 0 0 1 0 0 0 0 ...
 $ ResElectrocardio : int  2 0 0 2 0 2 2 2 2 0 ...
 $ MaxHeartRate     : int  160 141 105 121 140 142 142 170 154 161 ...
```



*Julio Alberto López Gómez*
*Juan Carlos Cubero, Coral del Val, Rocío Romero, Rafael Alcalá*

```
 $ ExcerciseInduced : int  0 0 1 1 0 1 1 0 0 0 ...
 $ OldPeak          : num  16 3 2 2 4 6 12 12 4 5 ...
 $ Slope            : int  2 1 2 1 1 2 2 2 2 2 ...
 $ MajorVessels     : int  0 0 1 1 0 1 1 2 3 0 ...
 $ Thal             : int  7 7 7 3 7 6 7 7 7 7 ...
 $ Class            : int  1 2 1 1 1 2 2 2 2 1 ...
> rangos_Heart <- apply(Heart, 2, range)
> rownames(rangos_Heart) <- c("min","max")
> rangos_Heart
```

Siendo la salida del último comando del anterior fragmento de código la que aparece en la siguiente imagen:

```
> rangos_Heart
    Age Sex ChestPainType RestBloodPressure SerumCholestoral FastingBloodSugar ResElectrocardio MaxHeartRate ExcerciseInduced OldPeak Slope MajorVessels Thal Class
min  29   0            1              94            126               0               0           71               0       0     1            0    3     1
max  77   1            4             200            564               1               2          202               1      62     3            3    7     2
```

Análogamente para Delta, se obtiene

```
> str(Delta)
> rangos_Delta <- apply(Delta, 2, range)
> rownames(rangos_Delta) <- c("min","max")
> rangos_Delta
    RollRate PitchRate CurrPitch CurrRoll DiffRollRate      Sa
min  -0.0208   -0.0077    -0.006   -0.050     -0.00150 -0.0021
max   0.0177    0.0108     0.041    0.051      0.00082  0.0022
```

Correspondiendo la salida de la función str(Delta) a la siguiente imagen:

```
> str(Delta)
'data.frame':   7128 obs. of  6 variables:
 $ RollRate    : num  -0.0023 0.0059 0.0046 -0.0076 -0.0073 0.0054 0.0061 -0.0038 0.0041 0.0058 ...
 $ PitchRate   : num  0.0003 -0.0006 0.0001 0.0012 0.0013 0.0027 0.0018 0.0008 0.0012 0.0012 ...
 $ CurrPitch   : num  0.022 0.014 0.011 0.01 0.01 0.017 0.019 0.015 0.013 0.014 ...
 $ CurrRoll    : num  -0.011 -0.005 0.015 0.001 -0.013 -0.022 -0.008 -0.014 -0.011 0.002 ...
 $ DiffRollRate: num  0.00005 0.00017 -0.00009 -0.0001 0.00006 0.00014 -0.00001 0.00001 0.00005 0.00004 ...
 $ Sa          : num  3e-04 -2e-04 -2e-04 2e-04 1e-04 -1e-04 -1e-04 1e-04 1e-04 -1e-04 ...
```

Como se puede observar tras la ejecución de la función str(), únicamente tenemos datos de tipos int y num. Los datos del dataset Heart son mayoritariamente del tipo int ya que no requieren de números decimales en su precisión, al haberse medido atributos discretos, mientras que los atributos del dataset Delta son de tipo num ya que se tratan de atributos intrínsecamente continuos, como tasas de balanceo, picos…

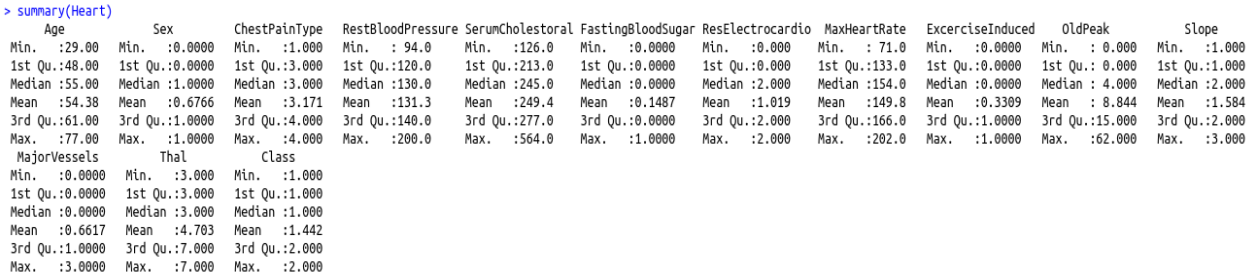
**A-1. Cálculo de media, desviación estándar, etc.**

Para calcular los estadísticos que se piden en este apartado, una forma sencilla y muy útil de obtener una panorámica de cada uno de los atributos que componen el dataset es utilizar la función summary(), la cual calcula para cada atributo una serie de estadísticos

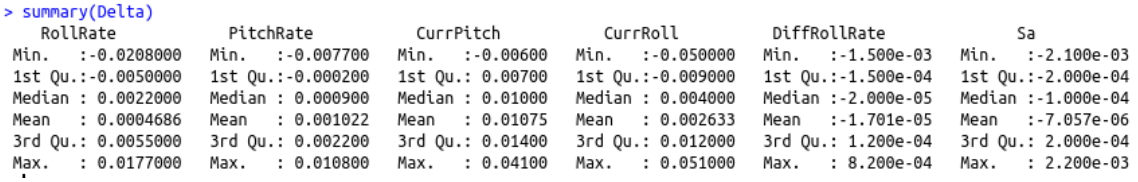
básicos. La aplicación de summary() sobre los datasets Heart y Delta_ail puede verse en el siguiente fragmento de código:

```
summary(Heart)
summary(Delta)
```

La salida de estos dos comandos puede apreciarse, respectivamente, en las siguientes imágenes:



```
> summary(Heart)
      Age             Sex         ChestPainType   RestBloodPressure SerumCholestoral FastingBloodSugar ResElectrocardio  MaxHeartRate   ExcerciseInduced    OldPeak          Slope
 Min.   :29.00   Min.   :0.0000   Min.   :1.000   Min.   : 94.0     Min.   :126.0     Min.   :0.0000    Min.   :0.000    Min.   : 71.0   Min.   :0.0000   Min.   : 0.000   Min.   :1.000
 1st Qu.:48.00   1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:120.0     1st Qu.:213.0     1st Qu.:0.0000    1st Qu.:0.000    1st Qu.:133.0   1st Qu.:0.0000   1st Qu.: 0.000   1st Qu.:1.000
 Median :55.00   Median :1.0000   Median :3.000   Median :130.0     Median :245.0     Median :0.0000    Median :2.000    Median :154.0   Median :0.0000   Median : 4.000   Median :2.000
 Mean   :54.38   Mean   :0.6766   Mean   :3.171   Mean   :131.3     Mean   :249.4     Mean   :0.1487    Mean   :1.019    Mean   :149.8   Mean   :0.3309   Mean   : 8.844   Mean   :1.584
 3rd Qu.:61.00   3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:140.0     3rd Qu.:277.0     3rd Qu.:0.0000    3rd Qu.:2.000    3rd Qu.:166.0   3rd Qu.:1.0000   3rd Qu.:15.000   3rd Qu.:2.000
 Max.   :77.00   Max.   :1.0000   Max.   :4.000   Max.   :200.0     Max.   :564.0     Max.   :1.0000    Max.   :2.000    Max.   :202.0   Max.   :1.0000   Max.   :62.000   Max.   :3.000
  MajorVessels        Thal           Class
 Min.   :0.0000   Min.   :3.000   Min.   :1.000
 1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:1.000
 Median :0.0000   Median :3.000   Median :1.000
 Mean   :0.6617   Mean   :4.703   Mean   :1.442
 3rd Qu.:1.0000   3rd Qu.:7.000   3rd Qu.:2.000
 Max.   :3.0000   Max.   :7.000   Max.   :2.000
```



```
> summary(Delta)
     RollRate            PitchRate           CurrPitch           CurrRoll           DiffRollRate              Sa
 Min.   :-0.0208000   Min.   :-0.007700   Min.   :-0.00600   Min.   :-0.050000   Min.   :-1.500e-03   Min.   :-2.100e-03
 1st Qu.:-0.0050000   1st Qu.:-0.000200   1st Qu.: 0.00700   1st Qu.:-0.009000   1st Qu.:-1.500e-04   1st Qu.:-2.000e-04
 Median : 0.0022000   Median : 0.000900   Median : 0.01000   Median : 0.004000   Median :-2.000e-05   Median :-1.000e-04
 Mean   : 0.0004686   Mean   : 0.001022   Mean   : 0.01075   Mean   : 0.002633   Mean   :-1.701e-05   Mean   :-7.057e-06
 3rd Qu.: 0.0055000   3rd Qu.: 0.002200   3rd Qu.: 0.01400   3rd Qu.: 0.012000   3rd Qu.: 1.200e-04   3rd Qu.: 2.000e-04
 Max.   : 0.0177000   Max.   : 0.010800   Max.   : 0.04100   Max.   : 0.051000   Max.   : 8.200e-04   Max.   : 2.200e-03
```

A la luz de los resultados provistos por la función summary(), se puede apreciar cómo para cada uno de los atributos de los datasets Heart y Delta, la función ha calculado su valor mínimo, el primer cuartil, la mediana y la media, el tercer cuartil y el valor máximo. Cabe destacar que todas estas medidas, son medidas de posición.

Para dar aún más valor a estos cálculos, a continuación se van a calcular otros estadísticos de dispersión para cada atributo, de manera que sea posible obtener una tabla que aglutine todos estos valores y que pueda ser consultada para obtener una visión global de los datos que se manejan. Los estadísticos calculados para los atributos de cada dataset son: varianza, desviación típica, coeficiente de variación y rango intercuartílico. Los cálculos de estos estadísticos para cada conjunto de datos se ven reflejados en el siguiente fragmento de código.

```
Var_Heart <- apply(Heart, 2, var)
Var_Delta <- apply(Delta, 2, var)
Sd_Heart <- apply(Heart, 2, sd)
Sd_Delta <- apply(Delta, 2, sd)
CV <- function(x) sd(x)/mean(x)
CV_Heart <- apply(Heart, 2, CV)
CV_Delta <- apply(Delta, 2, CV)
IQR_Heart <- apply(Heart, 2, IQR)
IQR_Delta <- apply(Delta, 2, IQR)
Estadisticos_Heart = rbind(Var_Heart,Sd_Heart,CV_Heart,IQR_Heart)
Estadisticos_Delta = rbind(Var_Delta,Sd_Delta,CV_Delta,IQR_Delta)
```

La vista de las matrices Estadísticos_Heart y Estadísticos_Delta, que aglutinan los estadísticos calculados para cada atributo del dataset, se obtiene mediante el siguiente fragmento de código, cuya salida se corresponde con las siguientes imágenes.

```
> Estadisticos_Heart
> Estadisticos_Delta
```

```
> Estadisticos_Heart
                Age       Sex ChestPainType RestBloodPressure SerumCholestoral FastingBloodSugar ResElectrocardio MaxHeartRate ExcerciseInduced    OldPeak     Slope MajorVessels
Var_Heart 82.3771570 0.2196360     0.9034844       320.2207180     2661.8358764        0.1270599        0.9959219  532.4557232        0.2222161 120.684486 0.3782389    0.8739389
Sd_Heart   9.0761863 0.4686534     0.9505180        17.8947120       51.5929828        0.3564546        0.9979589   23.0750021        0.4713980  10.985649 0.6150113    0.9348470
CV_Heart   0.1669169 0.6926800     0.2997530         0.1362374        0.2068764        2.3971571        0.9797479    0.1540089        1.4247872   1.242177 0.3883522    1.4127744
IQR_Heart 13.0000000 1.0000000     1.0000000        20.0000000       64.0000000        0.0000000        2.0000000   33.0000000        1.0000000  15.000000 1.0000000    1.0000000
               Thal     Class
Var_Heart 3.7694335 0.2476003
Sd_Heart  1.9415029 0.4975945
CV_Heart  0.4128571 0.3449818
IQR_Heart 4.0000000 1.0000000
```

```
> Estadisticos_Delta
               RollRate     PitchRate     CurrPitch      CurrRoll  DiffRollRate            Sa
Var_Delta 3.808610e-05 3.501566e-06 2.919551e-05 0.0002515876  4.561493e-08  9.172935e-08
Sd_Delta  6.171393e-03 1.871247e-03 5.403287e-03 0.0158615145  2.135765e-04  3.028685e-04
CV_Delta  1.316938e+01 1.831724e+00 5.024477e-01 6.0231673943 -1.255669e+01 -4.291942e+01
IQR_Delta 1.050000e-02 2.400000e-03 7.000000e-03 0.0210000000  2.700000e-04  4.000000e-04
```

### A-2. Gráficos que permitan visualizar los datos adecuadamente.

En esta sección, y una vez que los conjuntos de datos a estudiar han sido previamente analizados en la sección anterior, se van a presentar diferentes gráficos que ayudarán a ver de manera más global e incluso pormenorizada, las características de los datasets con los que se está trabajando. Por ello, se presentarán en primer lugar los gráficos realizados para el dataset heart, y a continuación, los gráficos obtenidos para el dataset Delta.

**Heart:** A pesar de la información suministrada anteriormente por las funciones str() y summary(), las cuales obtenían respectivamente un resumen de los atributos del conjunto de datos, sus valores, tipos y un resumen de las medidas de posición para cada uno de los atributos, se ha consultado el enlace[1] en el cual aparece el significado de las dos categorías del atributo clase, siendo:

- **Categoría 1:** ***Ausencia*** de enfermedad del corazón.
- **Categoría 2:** ***Presencia*** de enfermedad del corazón.

A continuación, se muestran los diferentes gráficos que se han realizado para una correcta visualización de los datos.

[1] https://archive.ics.uci.edu/ml/datasets/Statlog+%28Heart%29

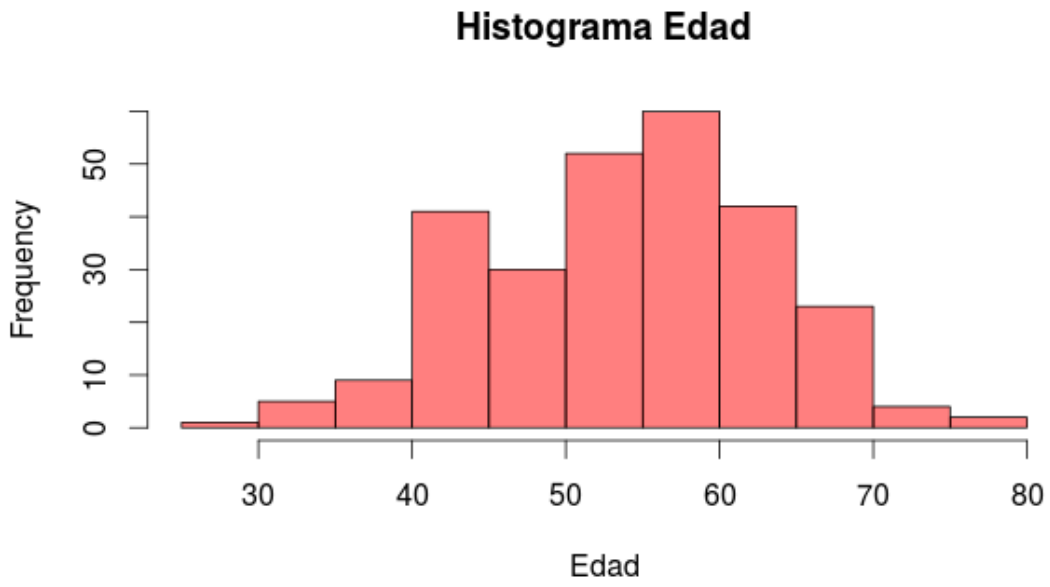
1. Histograma para la Variable Edad

Este histograma será útil para estudiar la edad de los individuos que han sido registrados. Si por ejemplo, todos estos individuos fuesen de edades menores de treinta años, los resultados obtenidos de este estudio no podrían ser extrapolados para pacientes de edad adulta o anciana, ya que es posible que en estas edades sea necesario otro estudio para ver si los factores que provocan una enfermedad del corazón son los mismos o cambian.

El código R utilizado para dibujar este histograma es:

```
hist(Heart$Age,
     main="Histograma Edad",
     xlab="Edad",
     col=rgb(1,0,0,0.5))
```

El histograma que genera este código se muestra a continuación.




Tal y como se puede apreciar en la imagen, el grueso de edad de los pacientes registrados se encuentra entre los cincuenta y los sesenta años, presentando dos colas a la izquierda y a la derecha significativas, que corresponden a los intervalos de pacientes entre cuarenta y cincuenta años y entre sesenta y setenta años, por lo que este conjunto de datos trabaja con datos de personas en edad adulta, luego los resultados extraídos han de extrapolarse a pacientes en estos intervalos de edades.

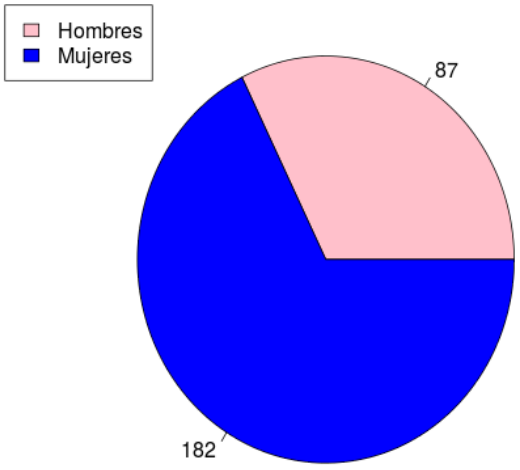


*Julio Alberto López Gómez*
*Juan Carlos Cubero, Coral del Val, Rocío Romero, Rafael Alcalá*

## 2. Gráfico que muestra el sexo de los pacientes registrados

Análogamente a como ocurría con la edad de los pacientes registrados, será importante conocer el sexo de los pacientes, ya que es posible observar diferencias con respecto a la presencia o ausencia de una enfermedad de corazón en ambos sexos. Estos datos también serán útiles para extrapolar de manera correcta los resultados de esta investigación, en función del número de hombres y de mujeres que hayan sido registrados. El código R que realiza este gráfico y el resultado de la ejecución del mismo, se muestran a continuación:

```
sexo <- table(Heart$Sex)
rownames(sexo) <-  c("Mujer", "Hombre")
sexo
pie(sexo, col = c("pink","blue"), labels = c(sexo[1],sexo[2]))
legend("topleft",c("Hombres","Mujeres"),bty="o",fill=c("pink","blue"))
```




Como se puede ver en el gráfico, de los pacientes registrados 87 de ellos son mujeres y 182 son hombres. Por lo que existe un mayor porcentaje de hombres registrados que de mujeres, lo cual repercutirá en los cálculos posteriores y en el proceso de clasificación, ya que un clasificador realizado solamente con datos de mujeres, clasificará mejor a éstas que un clasificador que se ha elaborado con datos de pacientes de ambos sexos, por lo que habrá que ser cuidadosos a la hora de extrapolar los resultados de esta investigación a según qué pacientes, tal y como se comentó anteriormente en el primer gráfico.

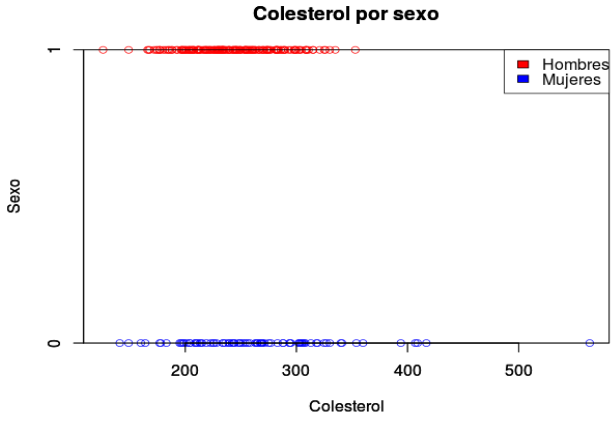
### 3. Gráfico que muestra los valores de colesterol para cada sexo

Uno de los principales factores de riesgo de contraer una enfermedad del corazón es el colesterol. Por este motivo, los siguientes gráficos muestran la distribución de la variable colesterol para hombres y mujeres. Para ello, se ha creado un dataframe con las variables colesterol y sexo y de ahí se han obtenido los dataframes colesterol_hombre y colesterol_mujeres, que se han representado en dos diagramas. Uno de ellos comparando los valores de colesterol de toda la muestra, separando hombres y mujeres, y el otro un diagrama de cajas donde podemos apreciar la distribución del colesterol para cada uno de los sexos, de forma más clara que en el primer gráfico.

El código R para realizar estos gráficos así como el resultado, se muestra a continuación.

```
sexo_colesterol <-as.data.frame(cbind(Heart$Sex,Heart$SerumCholestoral))
colnames(sexo_colesterol) = c("sexo","colesterol")
colesterol_mujeres <- sexo_colesterol[sexo_colesterol$sexo == 0,]
colesterol_hombres <- sexo_colesterol[sexo_colesterol$sexo == 1,]
rango_x <-
c(min(sexo_colesterol$colesterol),max(sexo_colesterol$colesterol))
rango_y <- c(0,1)
plot(colesterol_mujeres[,2],colesterol_mujeres[,1], main = "Colesterol
por sexo", xlim = rango_x, ylim = rango_y, xlab = "Colesterol", ylab =
"Sexo", col = c("blue"), lab = c(6,1,100))
par(new = TRUE)
plot(colesterol_hombres[,2],colesterol_hombres[,1], main = "Colesterol
por sexo", xlim = rango_x, ylim = rango_y, xlab = "Colesterol", ylab =
"Sexo", col = c("red"), lab = c(6,1,100))
legend("topright",   c("Hombres","Mujeres"), bty="o",
fill=c("red","blue"))

#Boxplot que clarifica el gráfico anterior
boxplot(colesterol_hombres$colesterol, colesterol_mujeres$colesterol,
main = "Colesterol por sexo", names = c("Mujeres", "Hombres"))
```







*Julio Alberto López Gómez*
*Juan Carlos Cubero, Coral del Val, Rocío Romero, Rafael Alcalá*

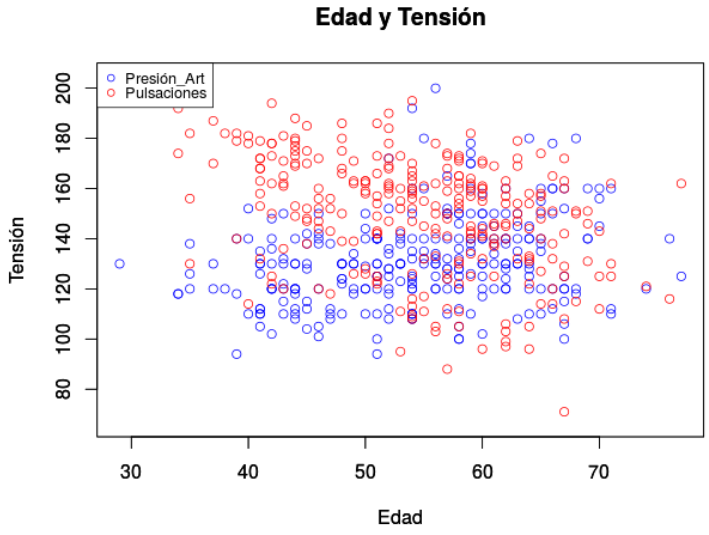
A la luz de los gráficos, podemos observar que la media del colesterol en los hombres es mayor que en las mujeres, así como el rango de colesterol en el que éstos se mueven, confirmando la tendencia de que son principalmente los hombres quienes tienen valores altos de colesterol, estando el colesterol del grueso de los hombres registrados entre doscientos y trescientos.

4. Gráfico comparativo de la Edad con respecto a la presión sanguínea y las pulsaciones máximas.

Otros factores importantes a tener en cuenta a la hora de presentar un riesgo alto de contraer una enfermedad del corazón son la presión sanguínea y las pulsaciones máximas. Por este motivo, se ha creado un gráfico que represente simultáneamente los valores de presión sanguínea y pulsaciones máximas para cada individuo, de manera que sea posible obtener una conclusión sobre cuáles son los márgenes de tensión y pulsaciones que pueden repercutir en el individuo haciendo que contraiga una enfermedad del corazón o cuáles son los valores de pulsaciones y tensión que presenta un individuo que padece del corazón.

El código R que realiza este gráfico y el resultado de ejecutar el mismo se muestran a continuación:

```
rango <- c(min(Heart$MaxHeartRate - 10),max(Heart$RestBloodPressure +
10))
plot(Heart$Age,Heart$RestBloodPressure, main = "Edad y Tensión", ylim
= rango, xlab = "Edad", ylab = "Tension", col = "blue")
par(new = TRUE)
plot(Heart$Age,Heart$MaxHeartRate, main = "Edad y Tensión", ylim =
rango, xlab = "Edad", ylab = "Tensión", col = "red")
legend("topleft",legend = c("Presión_Art","Pulsaciones"), cex=0.8, col
= c("blue","red"),pch=21)
```






*Julio Alberto López Gómez*
*Juan Carlos Cubero, Coral del Val, Rocío Romero, Rafael Alcalá*

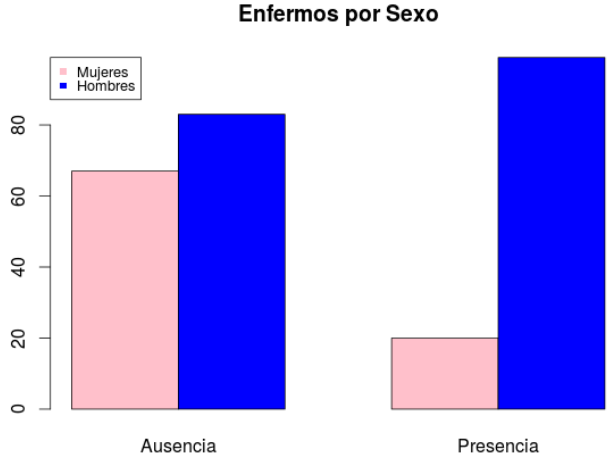
Una vez realizado el gráfico, podemos comprobar cómo la presión arterial se mantiene más o menos homogénea dentro de un intervalo de valores, mientras que los valores de las pulsaciones ocupan un rango mucho más amplio, presentando en ocasiones una gran diferencia con respecto a su valor de presión arterial, lo cual puede llevar a pensar, que cuando existe una diferencia muy grande entre la presión arterial y las pulsaciones, el paciente puede encontrarse en una situación propicia para contraer una enfermedad del corazón.

5. Gráfico que muestra los enfermos por sexo

En este último gráfico se presenta un diagrama de barras donde se puede comparar el número de hombres registrados que no presentaron una patología del corazón con respecto a las mujeres que no la presentaron así como el número de hombres registrados que sí presentaron una enfermedad del corazón, con respecto al número de mujeres que sí la presentaron. Este gráfico puede ser útil para determinar cuál es el sexo más castigado por este tipo de patologías o aquel que menos se cuida y que por este motivo contrae este tipo de enfermedades.

El código R que crea este gráfico así como el resultado del mismo, se muestran a continuación:

```
enfermos_sexo <- as.data.frame(cbind(Heart$Sex, Heart$Class))
colnames(enfermos_sexo) <- c("Sexo","Enfermo")
mujeres_enfermas <- enfermos_sexo[enfermos_sexo$Sexo == 0,]
mujeres_enfermas <-table(mujeres_enfermas)
hombres_enfermos <- enfermos_sexo[enfermos_sexo$Sexo == 1,]
hombres_enfermos <- table(hombres_enfermos)
Resultados <- rbind(mujeres_enfermas, hombres_enfermos)
barplot(Resultados, beside = TRUE, main = "Enfermos por Sexo", names =
c("Ausencia", "Presencia"), col =c("pink","blue"))
legend("topleft",legend  =  c("Mujeres","Hombres"),  cex=0.8,  col  =
c("pink","blue"), pch= 15)
```






*Julio Alberto López Gómez*
*Juan Carlos Cubero, Coral del Val, Rocío Romero, Rafael Alcalá*

A la vista del gráfico, se puede observar que existe un mayor número de hombres que no presentaron una enfermedad del corazón con respecto al número de mujeres, aunque es necesario recordar que se registraron más hombres que mujeres. Por otra parte, la diferencia entre los hombres que presentaron una enfermedad del corazón y las mujeres es abismal, pudiendo concluir que el sexo masculino es el más castigado por este tipo de patologías o el que menos se cuida y por tanto es más propenso a sufrir del corazón.

**Delta:** Para obtener una descripción más precisa de este Dataset, se ha accedido al enlace[2] para obtener una información más completa sobre este conjunto de datos. Los datos que aparecen en este dataset son fruto del control de los alerones de una aeronave F16, aunque cabe destacar que las variables y atributos presentes en él no son del dominio de los alerones, aunque sí que influyen en el comportamiento de éstos (motivo por el que se han capturado). Cabe destacar también que la variable a predecir en este conjunto de datos no es un valor absoluto sino una variación.

De manera análoga a como se hizo con el dataset Heart, a continuación se muestran los gráficos realizados para una mayor comprensión y estudio de los datos de Delta. Es importante destacar en este punto, que del mismo modo que el dataset Heart contiene información del corazón, la cual es más accesible al usuario, los datos obtenidos a priori del primer dataset posiblemente sean más ricos que los obtenidos en éste, ya que una base de aeronáutica sería ideal para poder estudiar relaciones entre las variables que se presentan en este dataset. Aun así, la labor de científico de datos también se ha realizado sobre este datset para estudiar a priori posibles relaciones, dependencias etc, que serán debidamente complementadas y profundizadas cuando se lleve a cabo el apartado de regresión del presente trabajo.

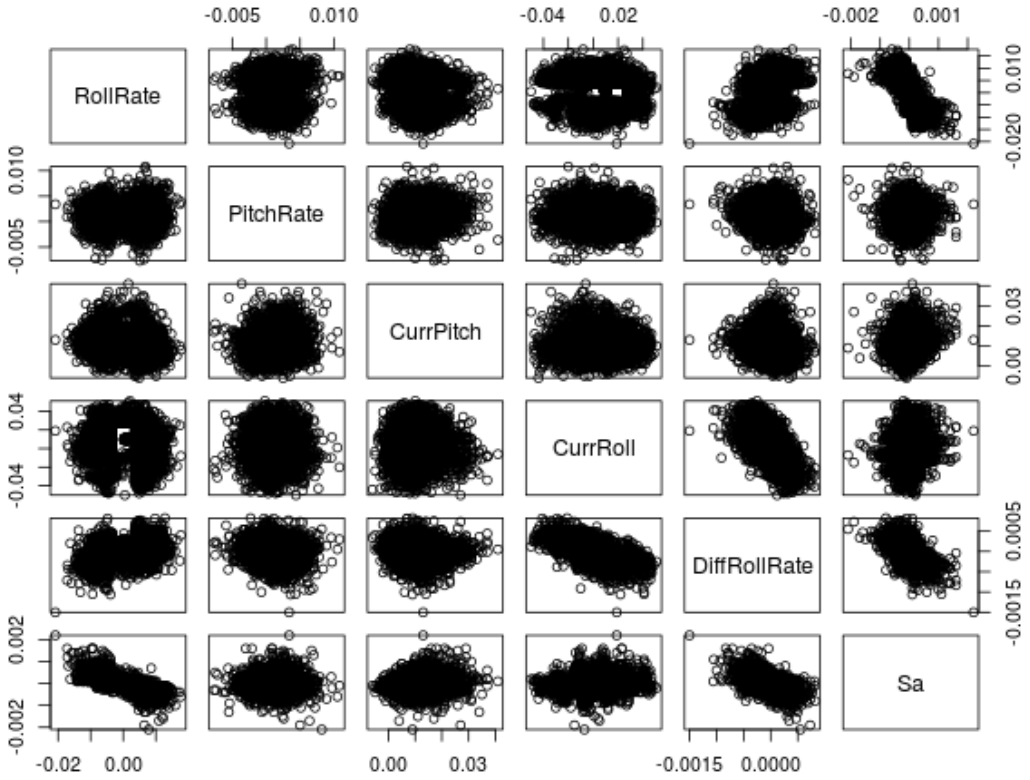
1. Scatterplot

Precisamente por el motivo aducido anteriormente de no tener una base de aeronáutica, y aprovechando que el dataset es un dataset estrecho, es decir, que contiene pocos atributos o variables, el primer gráfico que se realizará para detectar posibles relaciones y dependencias será un scatterplot que compare entre sí todas las variables del conjunto de datos Delta.

El código R realizado para ejecutar este gráfico así como el resultado, se muestra a continuación.

```
plot(Delta)
```

[2] http://sci2s.ugr.es/keel/dataset/data/regression/delta_ail-names.txt



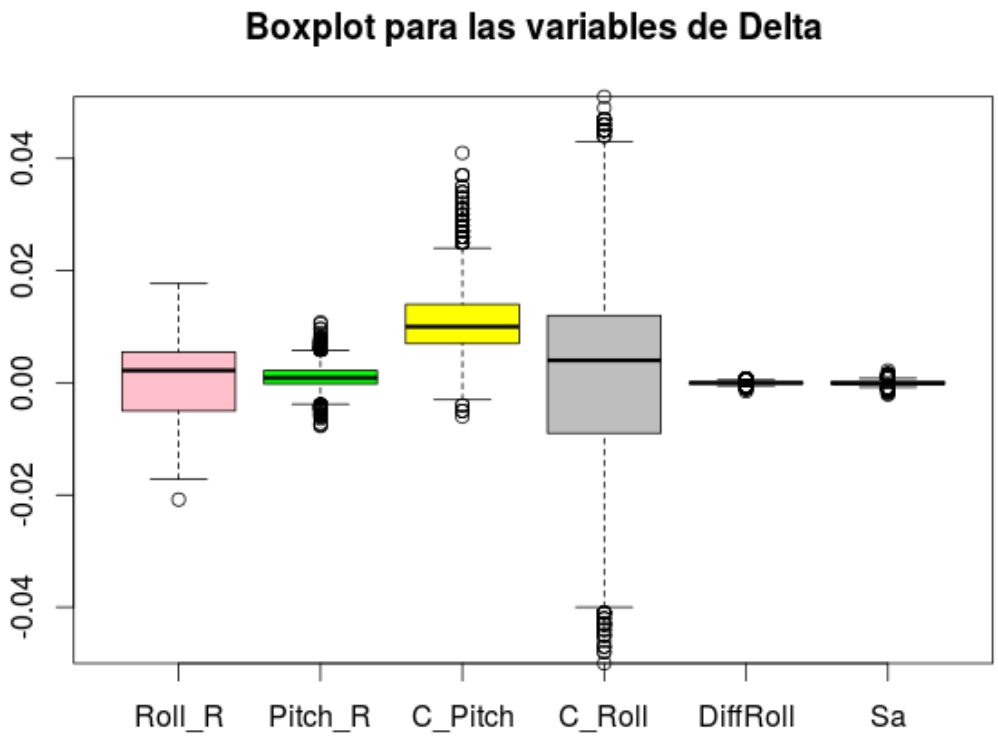

A priori, es posible observar que existe una relación entre la tasa de balanceo y la variable a predecir así como entre la diferencia de esta relación con la variable a predecir. Estas relaciones serán tenidas en cuenta a lo largo de este apartado, para graficarlas a posteriori.

## 2. Diagrama de Cajas

El segundo diagrama para seguir desmenuzando y familiarizándonos con datos de un dominio desconocido y que requiere de cierta formación o base, es un diagrama de cajas para conocer la distribución de cada una de las variables, de manera que después se puedan realizar otros gráficos y obtener conclusiones teniendo en cuenta las distribuciones de las variables que aporta este gráfico y las relaciones a priori que se observaron en el Scatterplot creado anteriormente.

El código R generado así como el resultado del gráfico, se muestran a continuación:

```
boxplot(Delta, names = c("Roll_R","Pitch_R", "C_Pitch", "C_Roll",
"DiffRoll","Sa"), main = "Boxplot para las variables de Delta", col =
c("pink","green","yellow", "grey", "white","blue"))
```




Como se puede ver en el gráfico, las dos últimas variables apenas pueden apreciarse, esto es debido a que tienen una escala diferente al resto. Tal vez un gráfico con las variables normalizadas fuera más representativo. De todos modos, podemos observar cómo se distribuyen las variables de balanceo y pico así como sus relaciones, llamando la atención que el valor medio de balanceo actual sea menor que el valor medio de pico alcanzado, mientras que a la hora de calcular su relación, el valor medio del ratio de balanceo sea mayor. Posiblemente esto sea debido al cálculo de esta relación.

### 3. Relación entre el balanceo y la variable a predecir

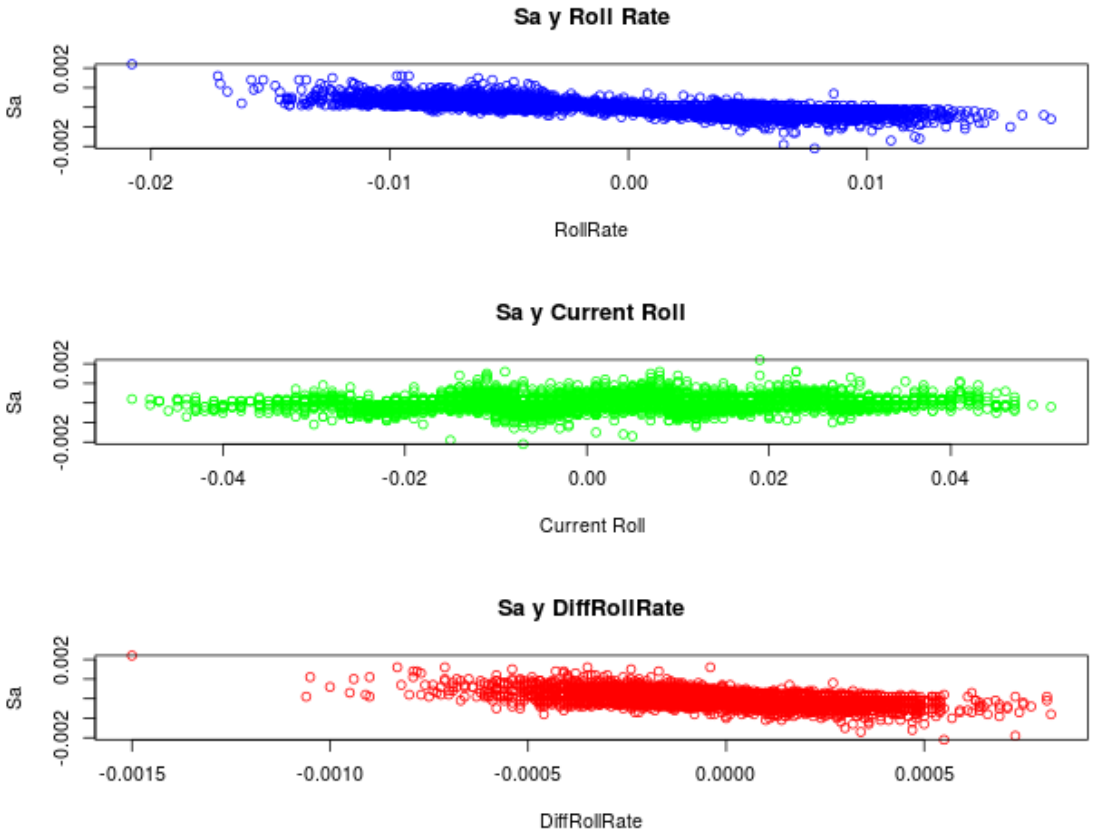
Tal y como se vio en el ScatterPlot, existe una relación entre el balanceo y la variable a predecir. Por este motivo, se ha decidido crear un gráfico que aglutine tres gráficos distintos que contendrán la relación entre cada uno de los valores de balanceo capturados con respecto a la variable a predecir. A continuación, se muestra el código R realizado para tal efecto y el gráfico generado.

```
par(mfrow = c(3,1))
plot(Delta$RollRate, Delta$Sa, main = "Sa y Roll Rate", xlab =
"RollRate", ylab = "Sa", col = "blue")
plot(Delta$CurrRoll,Delta$Sa, main = "Sa y Current Roll", xlab =
"Current Roll", ylab = "Sa", col = "green")
plot(Delta$DiffRollRate,Delta$Sa, main = "Sa y DiffRollRate", xlab =
"DiffRollRate", ylab = "Sa",col = "red")
```



*Julio Alberto López Gómez*
*Juan Carlos Cubero, Coral del Val, Rocío Romero, Rafael Alcalá*

```
par(mfrow = c(1,1))
```




Tal y como se observa en el gráfico, existe una clara relación inversa entre la variable a predecir, la tasa de balanceo y la diferencia del mismo, puesto que cuanto mayor es la tasa de balanceo o la diferencia, menor es la variable a predecir. Por su parte, el balanceo actual presenta un aspecto más homogéneo y aparentemente sin relación con respecto a la variable a predecir – por ese motivo, es por el que se calculan la tasa de balanceo y la diferencia, porque de ellas sí es posible obtener una relación –.

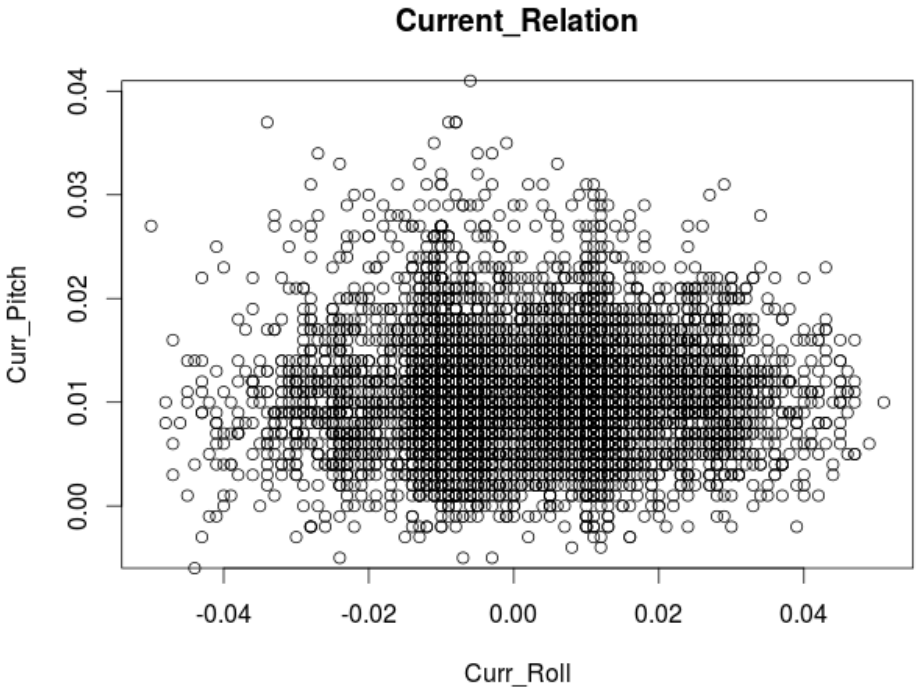
### 4. Gráfico para estudiar la relación entre el balanceo y el pico.

En el dataset Delta, dos son las variables principales a través de las cuáles se extraen las variables significativas que determinarán el valor de la variable objetivo. Estas variables son el balanceo (Roll) y el pico (Pitch). Por este motivo, se han construido dos gráficos para estudiar la relación entre el balanceo actual y el pico actual así como la relación entre la tasa de balanceo y la tasa de pico.

El código en R utilizado para crear estos gráficos así como su generación, se muestra a continuación:

***Relación entre Balanceo y Pico Actual.***

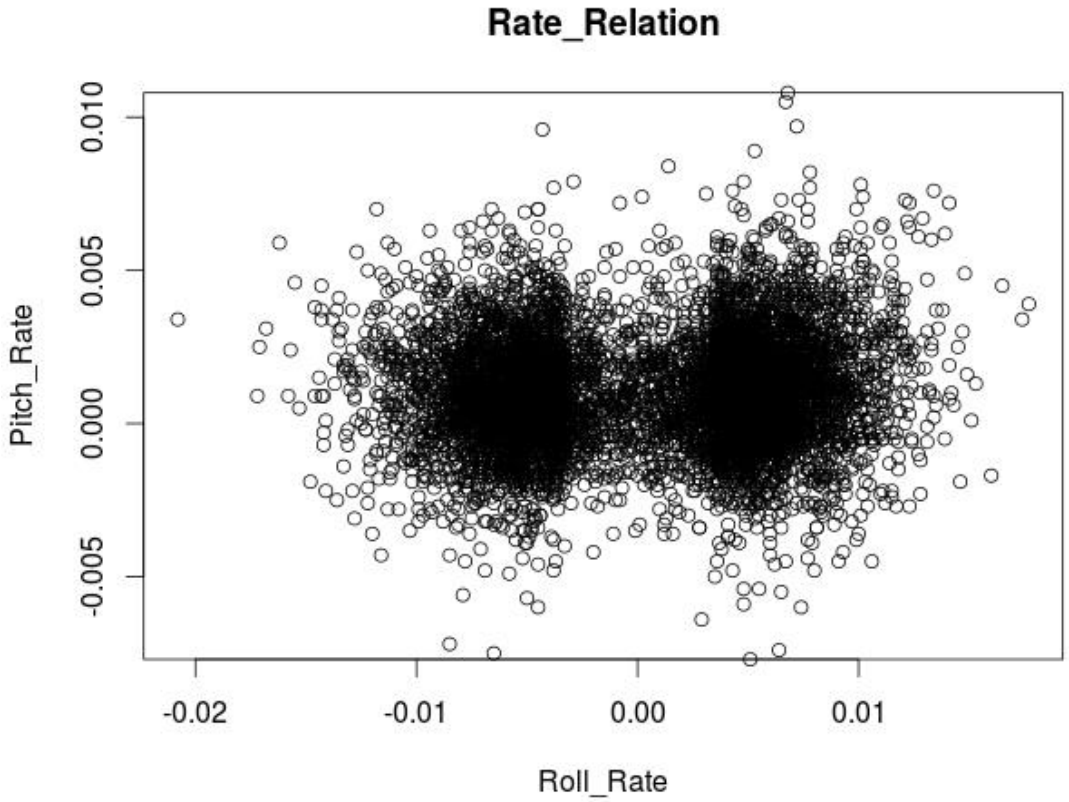
```
curr_roll <- Delta$CurrRoll
curr_pitch <- Delta$CurrPitch
Current_Relation <- as.data.frame(cbind(curr_roll, curr_pitch))
colnames(Current_Relation) <- c("Curr_Roll,", "Curr_Pitch")
Current_Relation <-Current_Relation[order(Current_Relation$Curr_Roll),]
plot(Current_Relation$Curr_Roll,Current_Relation$Curr_Pitch, xlab =
"Curr_Roll", ylab = "Curr_Pitch", main = "Current_Relation")
```




Este gráfico presenta la relación entre el balanceo y el pico actual, pudiendo ver que apenas existe una relación clara entre estas variables, aunque cabe destacar el intervalo entre -0.02 y 0.02 donde los picos parecen estar muy agrupados y variar muy poco.

***Relación entre Tasa de Balanceo y de Pico***

```
Roll_Rate <- Delta$RollRate
Pitch_Rate <- Delta$PitchRate
Rate_Relation <- as.data.frame(cbind(Roll_Rate, Pitch_Rate))
colnames(Rate_Relation) <- c("Roll_Rate", "Pitch_Rate")
Rate_Relation <- Rate_Relation[order(Rate_Relation$Roll_Rate),]
plot(Rate_Relation$Roll_Rate,Rate_Relation$Pitch_Rate, xlab =
"Roll_Rate", ylab = "Pitch_Rate", main = "Rate_Relation")
```




Este gráfico representa la relación entre las tasa de balanceo y pico. Como podemos ver, el gráfico presenta una estructura y una forma claramente llamativa, con dos núcleos a ambos lados del valor central 0 en la escala de balanceo, ya que dicha relación o tasa suele estar por encima o por debajo de este valor neutro. Estas dos relaciones sobre las que apenas podemos concluir nada en este estudio a priori, serán debidamente especificadas y explicitadas cuando se trabaje el apartado de regresión.

**A-3. Descripción del conjunto de datos a partir de los puntos anteriores.**

En este último apartado del primer punto del trabajo integrador, se describirán los datos con los que se trabajará posteriormente en los apartados de Regresión y Clasificación que contiene este trabajo. Esta definición de los datasets se llevará a cabo en función del estudio a priori que se ha venido haciendo a lo largo de esta sección con los dos conjuntos de datos. Cabe destacar que las conclusiones obtenidas son en principio superficiales, y que serán debidamente probadas o refutadas a medida que se realicen los apartados de regresión y clasificación y se apliquen diferentes algoritmos y modelos que terminarán de definir, probar y ampliar la información extraída de este primer estudio.

**Heart:** Estas son las principales conclusiones y la definición obtenida de este dataset tras su estudio preliminar:

1. Se trata de un data frame R.
    a. Es un data frame corto, pues tan solo posee 269 observaciones.
    b. Es un data frame relativamente ancho, pues posee 14 atributos o variables.
    c. La gran mayoría de estas variables son de tipo int, al haberse medido valores intrínsecamente discretos y que no requieren de una precisión grande.
2. Los estadísticos principales de cada variable son:

```
> Estadisticos_Heart
               Age       Sex ChestPainType RestBloodPressure SerumCholestoral FastingBloodSugar ResElectrocardio MaxHeartRate ExcerciseInduced    OldPeak      Slope MajorVessels
Var_Heart 82.3771570 0.2196360     0.9034844       320.2207180     2661.8358764         0.1270599        0.9959219  532.4557232        0.2222161 120.684486 0.3782389    0.8739389
Sd_Heart   9.0761863 0.4686534     0.9505180        17.8947120       51.5929828         0.3564546        0.9979589   23.0750021        0.4713980  10.985649 0.6150113    0.9348470
CV_Heart   0.1669169 0.6926800     0.2997530         0.1362374        0.2068764         2.3971571        0.9797479    0.1540089        1.4247872   1.242177 0.3883522    1.4127744
IQR_Heart 13.0000000 1.0000000     1.0000000        20.0000000       64.0000000         0.0000000        2.0000000   33.0000000        1.0000000  15.000000 1.0000000    1.0000000
               Thal     Class
Var_Heart 3.7694335 0.2476003
Sd_Heart  1.9415029 0.4975945
CV_Heart  0.4128571 0.3449818
IQR_Heart 4.0000000 1.0000000
```

3. La edad de los individuos registrados se encuentra principalmente entre los cuarenta y los setenta años.
4. En los individuos registrados existe un mayor porcentaje de hombres que de mujeres
5. El colesterol de los hombres registrados es mayor que el colesterol que presentan las mujeres registradas
6. Los hombres registrados están más afectados por enfermedades del corazón que las mujeres registradas.

**Delta:** Estas son las principales conclusiones y la definición obtenida de este dataset tras su estudio preliminar:

1. Se trata de un data frame R:
    a. Es un data frame relativamente largo, pues tiene 7128 filas.
    b. Es un data frame estrecho, ya que solo tiene 7 variables.
    c. La mayoría de estas variables son de tipo num, pues en este caso las variables de este dominio son números de punto flotante que requieren gran precisión.
2. Los estadísticos principales de cada variable son:

```
> Estadisticos_Delta
              RollRate    PitchRate    CurrPitch     CurrRoll  DiffRollRate           Sa
Var_Delta 3.808610e-05 3.501566e-06 2.919551e-05 0.0002515876  4.561493e-08  9.172935e-08
Sd_Delta  6.171393e-03 1.871247e-03 5.403287e-03 0.0158615145  2.135765e-04  3.028685e-04
CV_Delta  1.316938e+01 1.831724e+00 5.024477e-01 6.0231673943 -1.255669e+01 -4.291942e+01
IQR_Delta 1.050000e-02 2.400000e-03 7.000000e-03 0.0210000000  2.700000e-04  4.000000e-04
```

3. Existe una relación entre el balanceo y la variable a predecir, tal y como se vio en el gráfico que muestra las diferentes medidas de balanceo con respecto a la variable a predecir.
4. Parece existir una relación entre las dos variables principales del dataset: Balanceo y Pico. Esto será probado y demostrado o refutado en el apartado de regresión.

## 2. Regresión

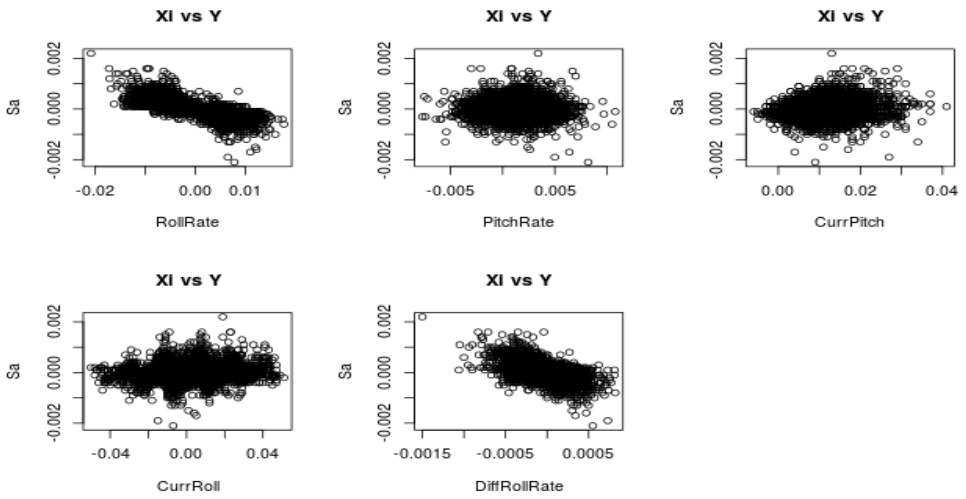
**En este apartado el estudiante debe utilizar el dataset$_R$ asignado, en este caso Delta_ail, para realizar lo siguiente:**

**R-1. Utilizar el algoritmo de regresión lineal simple sobre cada regresor (variable de entrada) para obtener los modelos correspondientes. Si el dataset R asignado incluye más de 5 regresores, seleccione de manera justificada los 5 que considere más relevantes. Una vez obtenidos los modelos, elegir el que considere más adecuado para su conjunto de datos según las medidas de calidad conocidas.**

En primer lugar, y tal y como se vio en el primer apartado del presente trabajo, el dataset Delta_ail contiene cinco variables o regresores, por lo que se aplicará esta tarea sobre cada uno de ellos. No obstante, cabe destacar que en caso de que existieran más de cinco regresores, sería necesario estudiar cuáles son aquellos más importantes para nuestro estudio. Por este motivo, se deja a modo de anotación, el código necesario para estudiar y evaluar cuáles serían los regresores sobre los que aplicar el algoritmo de regresión lineal simple en caso de que el dataset presentara más de cinco variables o atributos. Para ello, se dibujará cada variable o regresor con respecto a la variable a predecir, parar observar posibles relaciones que determinen sobre qué regresores se construirán los modelos de regresión.

El código R para visualizar la relación entre los regresores y la variable a predecir, así como dicha visualización, se muestra a continuación.

```
plotY<-function (x,y) {
  plot(Delta[,y]~Delta[,x], xlab=paste(names(Delta)[x])
       , ylab=names(Delta)[y],main="Xi vs Y")
}
par(mfrow=c(2,3))
#Dibujamos cada Xi con respecto a Y
x <-sapply(1:(dim(Delta)[2]-1), plotY, dim(Delta)[2])
par(mfrow=c(1,1))
```



Una vez estudiadas todas las posibles relaciones entre los regresores que presenta el dataset y la variable a predecir, y una vez identificados aquellos más significativos, se aplicará el algoritmo de regresión lineal simple sobre cada regresor. Para ello, se ha construido una función que dadas dos variables x e y así como un conjunto de datos de prueba, calcula el modelo de regresión de y sobre x, el intervalo de confianza del modelo, el error del mismo y realiza las predicciones de la variable y con el conjunto de datos que ha recibido como argumento, calculando finalmente el error cometido en la predicción.

Esta función, cuyo código R se muestra a continuación, devuelve una lista con el resumen del modelo de regresión calculado, el intervalo de confianza, el RMSE del modelo de regresión calculado y el RMSE cometido durante la predicción.

```
### Función que calcula el modelo de regresión lineal para una variable#
respecto de otra.                                                      ##
##                                                                     ##
##          Argumentos -> Variables x (regresor) e y (variable a       ##
predecir) y  test(Conjunto de Prueba para predecir)                    ##
##          Salida -> Lista[                                           ##
##                          Modelo                                     ##
##                          Resumen del Modelo                         ##
##                          Intervalo de Confianza                     ##
##                          RMSE del Modelo                            ##
##                          RMSE en las predicciones realizadas        ##
##                          ]                                          ##
## -.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-##
Construir_MRegresion_Lineal <- function(x,y,test){                     ##
#Se construye el modelo de regresión                                   ##
modelo <- lm(y~x, data = Delta)                                        ##
#Se calcula el intervalo de confianza                                  ##
intervalo <- confint(modelo)                                           ##
#Cálculo del RMSE del modelo                                           ##
RMSE_modelo <- sqrt(sum(modelo$residuals^2)/length(modelo$residuals)) ##
#Realización de predicciones                                           ##
yprime <- predict(modelo, data.frame(x = test))                        ##
#Cálculo del RMSE en las predicciones                                  ##
RMSE_yprime <- sqrt(sum(abs(Delta$Sa - yprime)^2)/length(yprime))      ##
lista <- list(modelo,summary(modelo), intervalo, RMSE_modelo, RMSE_yprime)##
                                                                       ##
return(lista)                                                          ##
}                                                                      ##
########################################################################
```

Una vez construida la función, se aplicará sobre cada uno de los regresores para obtener el modelo de regresión lineal simple asociado a cada uno de ellos con respecto a la variable a predecir. Como conjunto de prueba para cada uno de estos modelos, se seleccionará un conjunto aleatorio de valores que estén dentro del rango de la variable

independiente o variable x. A continuación, se muestra el código R para cada uno de los modelos de regresión lineal calculados, así como el resultado de las medidas de calidad conocidas para cada uno de ellos.

**Modelo de regresión para RollRate**

```
aleatorios_RollRate <-
runif(Delta$RollRate,rangos_Delta[1,1],rangos_Delta[2,1])
fit_RollRate <-
Construir_MRegresion_Lineal(Delta$RollRate,Delta$Sa,aleatorios_RollRat
e)
fit_RollRate
```

Este modelo explica un 59.5% de variabilidad, lo que indica que existe una relación entre el tasa de Balanceo y la variable a predecir. Por otra parte, el error cometido por el modelo es de 0.00019 por lo que es bastante preciso, mientras que el error para el conjunto de prueba es de 0.00052, lo que incrementa significativamente éste en el orden de la diezmilésima.

**Modelo de regresión para PitchRate**

```
aleatorios_PitchRate <-
runif(Delta$RollRate,rangos_Delta[1,2],rangos_Delta[2,2])
fit_PitchRate <-
Construir_MRegresion_Lineal(Delta$PitchRate,Delta$Sa,aleatorios_PitchR
ate)
fit_PitchRate
```

Este modelo construido para la variable PitchRate explica tan solo un 0.02% de variabilidad, por lo que podemos concluir que existe una relación mucho más fuerte o importante entre la variable a predecir y la tasa o ratio de balanceo que entre la variable a predecir y la tasa de pico. Por su parte, el error cometido por el modelo es muy pequeño y existe muy poca diferencia entre el error cometido en las predicciones(0.000303) y el error cometido por el modelo(0.000302). Este aspecto llama mucho la atención, es decir, el hecho de que un modelo que explica tan poca variabilidad presente un error tan pequeño.

**Modelo de regresión para CurrPitch**

```
aleatorios_CurrPitch <-
runif(Delta$RollRate,rangos_Delta[1,3],rangos_Delta[2,3])
fit_CurrPitch <-
Construir_MRegresion_Lineal(Delta$CurrPitch,Delta$Sa,aleatorios_CurrPi
tch)
fit_CurrPitch
```

Si se ejecuta el código R proporcionado anteriormente, se podrá comprobar que este modelo tan solo explica un 2.92% de variabilidad, lo cual es indicativo de que existe una relación muy débil entre el pico actual y la variable a predecir. Por su parte, el error cometido por el modelo es de 0.00029, mientras que el error cometido en las predicciones es de 0.00033, por lo que existe una pequeña diferencia entre ellos.

**Modelo de regresión para CurrRoll**

```
aleatorios_CurrRoll <-
runif(Delta$RollRate,rangos_Delta[1,4],rangos_Delta[2,4])
fit_CurrRoll <-
Construir_MRegresion_Lineal(Delta$CurrRoll,Delta$Sa,aleatorios_CurrRol
l)
fit_CurrRoll
```

Este modelo tan solo explica un 0.3% de variabilidad, por lo que de forma análoga a como sucedía con la variable CurrPitch, la relación entre la variable a predecir y el balanceo actual es muy débil o prácticamente inexistente. Por su parte, el error cometido por el modelo es de 0.000302 mientras que el error cometido en las predicciones realizadas con el conjunto de prueba suministrado es de 0.000304.

**Modelo de regresión para DiffRollRate**

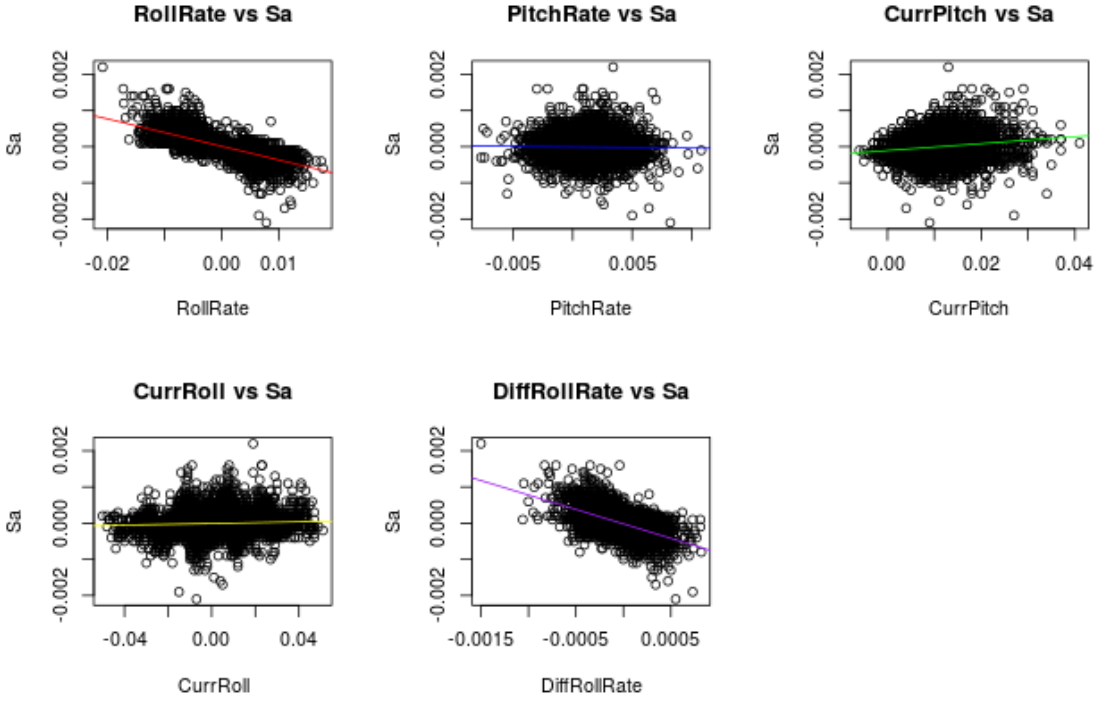
```
aleatorios_DiffRollRate <-
runif(Delta$RollRate,rangos_Delta[1,5],rangos_Delta[2,5])
fit_DiffRoll <-
Construir_MRegresion_Lineal(Delta$DiffRollRate,Delta$Sa,aleatorios_Dif
fRollRate)
fit_DiffRoll
```

Este modelo explica un 31.8% de variabilidad, por lo que sí podemos afirmar que existe cierta relación entre la variable a predecir y la diferencia de las tasas de balanceo. Por su parte, el error cometido por este modelo es de 0.000250 mientras que el error cometido en la predicción es de 0.000669.

Una vez creados todos los modelos de regresión lineal, podemos concluir que en principio, el mejor es el primero, que explica casi un 60% de variabilidad y que define una relación entre la tasa de balanceo y la variable a predecir. No obstante, en la siguiente sección se trabajará para obtener mejores modelos mediante modelos multivariantes, interacciones y no linealidad.

Por último, antes de pasar a la siguiente sección del apartado de regresión, se proporciona el código R que realiza un gráfico comparativo de todos los modelos de regresión calculados con su ajuste.

```
par(mfrow = c(2,3))
plot(Delta$RollRate,Delta$Sa, main = "RollRate vs Sa", xlab =
"RollRate", ylab = "Sa")
abline(fit_RollRate, col = "red")
plot(Delta$PitchRate,Delta$Sa, main = "PitchRate vs Sa", xlab =
"PitchRate", ylab = "Sa")
abline(fit_PitchRate, col = "blue")
plot(Delta$CurrPitch,Delta$Sa, main = "CurrPitch vs Sa", xlab =
"CurrPitch", ylab = "Sa")
abline(fit_CurrPitch, col = "green")
plot(Delta$CurrRoll,Delta$Sa, main ="CurrRoll vs Sa", xlab =
"CurrRoll", ylab = "Sa")
abline(fit_CurrRoll, col = "yellow")
plot(Delta$DiffRollRate,Delta$Sa, main ="DiffRollRate vs Sa", xlab =
"DiffRollRate", ylab = "Sa")
abline(fit_DiffRoll, col = "purple")
```




**R-2. Utilizar el algoritmo para regresión lineal múltiple. Justificar adecuadamente si el modelo obtenido aporta mejoras respecto al modelo elegido en el paso anterior (en este apartado tenga también en cuenta la consideración de posibles interacciones y no linealidad).**

Con vistas a mejorar los modelos de regresión calculados en el apartado anterior, en esta sección se utilizará el algoritmo de regresión lineal múltiple, aplicando interacciones y no linealidades, con el objetivo de mejorar los modelos del apartado anterior.

En primer lugar, se han realizado los siguientes modelos de regresión multivariante que se consideran a continuación:

**Modelo de regresión para todas las variables**

```
fit_all <- lm(Delta$Sa~.,data = Delta)
summary(fit_all)$adj.r.squared
```

Este modelo de regresión aglutina los cinco regresores del dataset y explica un 67.7% de variabilidad, convirtiéndose así en el mejor modelo construido hasta el momento.

**Modelo de Regresión para la relación entre el Balanceo y Pico.**

```
fit_rate <- lm(Delta$Sa ~ Delta$RollRate+Delta$PitchRate, data =
Delta)
summary(fit_rate)$adj.r.squared
```

Como ya se describió en el primer apartado del presente trabajo, el Dataset Delta_ail contiene dos variables principales a partir de las cuales se extraen el resto de variables o regresores que aparecen en el dataset. Estas variables son el balanceo y el pico, y por este motivo se ha construido un modelo de regresión multivariante con dichas variables, que explica un 59.7% de variabilidad.

**Modelo de regresión para la relación entre el balanceo y el pico actuales.**

```
fit_curr <- lm(Delta$Sa ~ Delta$CurrRoll+Delta$CurrPitch, data =
Delta)
summary(fit_curr)$adj.r.squared
```

Análogamente al anterior, se ha construido un modelo de regresión multivariante que aglutine el balanceo y el pico actuales para una aeronave F16. Sin embargo, este modelo solo ha conseguido explicar un 3,2% de variabilidad.

**Modelo de regresión para el Ratio de Balanceo y la Diferencia de éste**

```
fit_roll <- lm(Delta$Sa ~ Delta$RollRate + Delta$DiffRollRate, data =
Delta)
summary(fit_roll)$adj.r.squared
```

Matemáticamente, es fácil deducir la posibilidad de que exista una relación entre el ratio de balanceo y la diferencia de éste, que intuitivamente tomará el Ratio de balanceo actual y el anterior para efectuar su cálculo. Por este motivo se ha construido este modelo de regresión que explica un 64% de variabilidad.

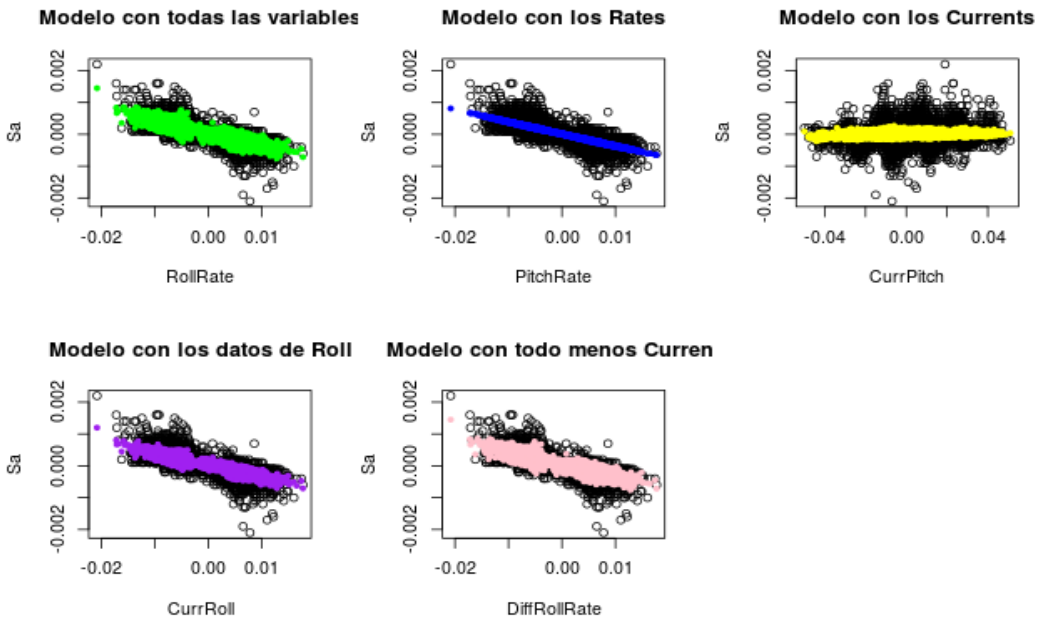
De los modelos obtenidos anteriormente, el mejor es el primero de ellos. Este modelo es un modelo de regresión multivariante que aglutina todas las variables y que explica un 67.7% de variabilidad. Para tratar de reducir la complejidad de este modelo, se han eliminado las variables CurrPitch y CurrRoll, ya que como se vio anteriormente en el tercer modelo calculado, no son de gran importancia. El modelo resultado de esta eliminación es:

```
fit_sinCurr <- lm(Delta$Sa ~.-Delta$CurrPitch-Delta$CurrRoll, data =
Delta)
summary(fit_roll)$adj.r.squa
```

Y explica un 64% de variabilidad, por lo que no mejora al primero que emerge como el mejor modelo calculado hasta el momento.

Una panorámica de los modelos de regresión multivariantes calculados, puede verse en el gráfico que aparece a continuación, resultado de ejecutar el siguiente código R:

```
par(mfrow = c(2,3))
plot(Delta$RollRate,Delta$Sa, main = "Modelo con todas las variables",
xlab = "RollRate", ylab = "Sa")
points(Delta$RollRate,fitted(fit_all),col="green",pch=20)
plot(Delta$RollRate,Delta$Sa, main = "Modelo con los Rates", xlab =
"PitchRate", ylab = "Sa")
points(Delta$RollRate,fitted(fit_rate),col="blue",pch = 20)
plot(Delta$CurrRoll,Delta$Sa, main = "Modelos con los Currents", xlab
= "CurrPitch", ylab = "Sa")
points(Delta$RollRate,fitted(fit_curr),col="yellow",pch=20)
plot(Delta$RollRate,Delta$Sa, main ="Modelo con los datos de Roll",
xlab = "CurrRoll", ylab = "Sa")
points(Delta$RollRate,fitted(fit_roll),col="purple",pch = 20)
plot(Delta$RollRate,Delta$Sa, main ="Modelo con todo menos Current",
xlab = "DiffRollRate", ylab = "Sa")
points(Delta$RollRate,fitted(fit_sinCurr),col="pink",pch = 20)
```






*Julio Alberto López Gómez*
*Juan Carlos Cubero, Coral del Val, Rocío Romero, Rafael Alcalá*

Una vez realizados los modelos multivariantes, se va a proceder a tratar de mejorarlos empleando interacciones y no linealidad. Con este objetivo, se han construido los siguientes modelos:

**Modelo con interacciones entre las variables de Roll.**

```
fit_int <- lm(Delta$Sa ~
Delta$RollRate*Delta$DiffRollRate*Delta$CurrRoll, data = Delta)
summary(fit_int)$adj.r.squared
```

Si interaccionan las variables RollRate, CurrRoll y DiffRollRate, se obtiene el modelo que aparece en el código R anterior y que explica un 67.8% de variabilidad.

**Modelo con interacciones entre las variables de Rate**

```
fit_int2 <- lm(Delta$Sa ~ Delta$RollRate*Delta$PitchRate)
summary(fit_int2)$adj.r.squared
```

Al interaccionar las variables RollRate y PitchRate que como se vio anteriormente presentaban cierta relación, obtenemos este modelo que explica un 59.7% de variabilidad.

**Modelo no lineal para las variables Rate**.

```
fit_nl <- lm(Delta$Sa ~ Delta$RollRate + Delta$RollRate^2 +
Delta$PitchRate + Delta$PitchRate^2, data = Delta)
summary(fit_nl)$adj.r.squared
```

Para tratar de mejorar el tanto por ciento de variabilidad que explicaban los modelos que relacionaban las variables RollRate y PitchRate, se ha construido este modelo no lineal con la expresión: $RollRate + RollRate^2 + PitchRate + PitchRate^2$. Este modelo vuelve a explicar un 59.7% de variabilidad, luego no ha sido posible mejorarlo mediante el modelo construido.

**Modelo no lineal para las variables Roll**

```
fit_nl1 <- lm(Delta$Sa ~ Delta$RollRate + Delta$RollRate^2 +
Delta$DiffRollRate + Delta$DiffRollRate^2 + Delta$CurrRoll +
Delta$CurrRoll^2, data = Delta)
summary(fit_nl1)$adj.r.squared
```

Con el objetivo de mejorar los modelos construidos con las variables de Roll, se ha construido el modelo con la expresión: $RollRate + RollRate^2 + DiffRollRate + DiffRollRate^2 + CurrRoll + CurrRoll^2$ . Así, se ha obtenido un modelo que explica un 64.4% de variabilidad, no logrando mejorar el 67% que obtuvimos anteriormente.

**Modelo polinómico de grado 10 para RollRate**

```
fit_nl2 <- lm(Delta$Sa ~ poly(Delta$RollRate,10))
summary(fit_nl2)
```

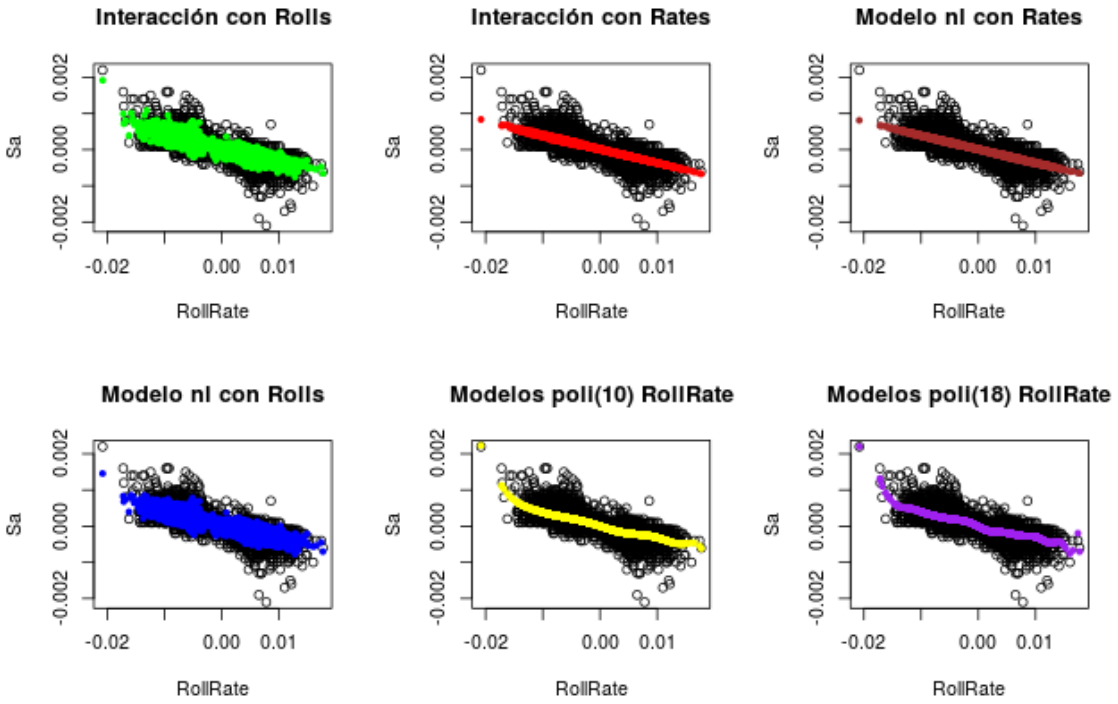
Siendo RollRate la variable que más relación presentaba con respecto a la variable a predecir, se ha elaborado un modelo de regresión polinómico de grado diez para observar si este modelo mejora los anteriores. El modelo calculado mediante el código R mostrado explica solo un 60.71% de variabilidad

**Modelo polinómico de grado 18 para RollRate**

```
fit_nl3 <- lm(Delta$Sa ~ poly(Delta$RollRate,18))
summary(fit_nl3)
```

Tratando de solucionar las carencias del modelo anterior, se ha construido un nuevo modelo polinómico, esta vez de grado 18 para observar si merece la pena aumentar el grado de complejidad del modelo debido a la ganancia obtenida. Sin embargo, este modelo solo explica el 60.78% de variabilidad, por lo que no merece la pena aumentar la complejidad del modelo ya que la ganancia es mínima.

En resumen, el siguiente gráfico muestra el ajuste de todos los modelos con interacciones y no linealidad, vistos en esta sección.




**Conclusión:** Tras analizar todos los modelos de regresión construidos en los dos primeros apartados de esta sección de regresión, es posible llegar a la conclusión de que el mejor modelo construido es aquel en el que se produce la interacción entre todas

las variables Roll y que representa un 67.8% de variabilidad. Por otra parte, la diferencia entre este modelo y el modelo lineal multivariante que aglutina todas las variables es irrisoria, de modo que se podría elegir cualquiera de estos dos modelos.

Es en este punto cuando es preciso deliberar qué es más conveniente, si un modelo lineal, mucho más simple, pero que aglutina todas las variables del problema (aunque para éste caso concreto solo son cinco), o un modelo con interacciones, algo más complejo, pero que aglutina las únicas variables decisivas en este problema.

Por otra parte, y de manera muy similar a los modelos que explican casi el 68% de variabilidad, se encuentran los modelos que aglutinan todas las variables excepto las variables Current, el modelo multivariante que integra las variables RollRate y DiffRollRate y el modelo no lineal construido para las variables de Roll (todos ellos explican un 64% de variabilidad). Ante esta situación, será necesario plantearse si merece la pena reducir la complejidad del modelo, ya que el modelo multivariante que integra las variables RollRate y DiffRollRate es mucho más simple, pero perder un 4% de variabilidad.

**R-3. Aplicar el algoritmo k-NN para regresión.**

En esta sección se aplicará el algoritmo k-NN para cada uno de los modelos construidos anteriormente, calculando su RMSE y haciendo una comparativa entre todos ellos.

En primer lugar, será necesario cargar la función kknn() que permitirá aplicar este algoritmo para el conjunto de datos de regresión Delta que se ha utilizado a lo largo del trabajo. Esto se consigue con el siguiente código R:

```
require("kknn")
```

Una vez que esta función ha sido cargada, se aplicará el algoritmo k-NN para los modelos calculados anteriormente, es decir, se aplicará en primer lugar a los modelos de regresión lineal construidos, después a los modelos de regresión multivariante y por último a los modelos de regresión multivariante que presentan interacciones y no linealidad.

**Modelos de regresión lineal**

En este apartado se aplicará el algoritmo k-NN para cada uno de los modelos de regresión lineal aplicados sobre cada uno de los regresores del conjunto de datos. El código R que permite aplicar este algoritmo, se muestra a continuación:

```
#Ejecutamos knn de Sa sobre RollRate
knn_fit_RollRate <- kknn(Delta$Sa ~ Delta$RollRate, Delta, Delta)
#Hallamos los valores predichos
yprime <- knn_fit_RollRate$fitted.values
#Cálculo del RMSE
```

```
RMSE_RollRate <- sqrt(sum((Delta$Sa - yprime)^2)/length(yprime))
RMSE_RollRate
```

En el fragmento de código presentado anteriormente, se muestra la aplicación del algoritmo knn sobre el regresor RollRate, sobre el cual se aplica el algoritmo, se realiza la predicción y se calcula el RMSE.

Por simplicidad y reducción de la extensión de la memoria, no se incluirá el código R de la aplicación del algoritmo a cada uno de los regresores, ya que es análogo al que se ha aportado anteriormente.

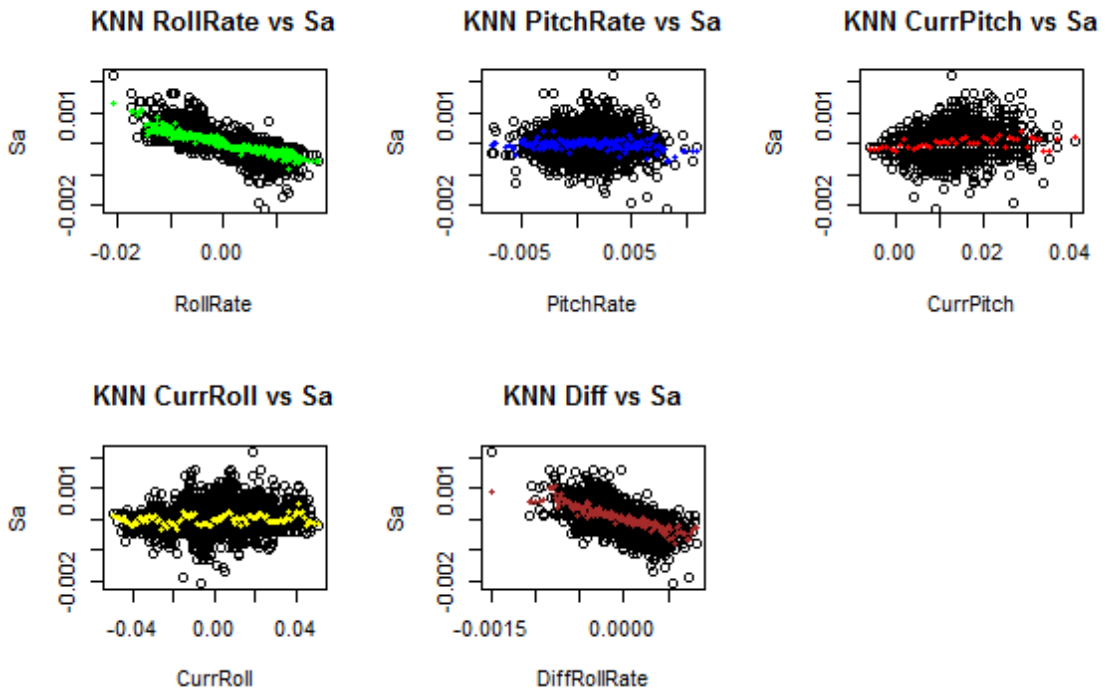
La siguiente tabla comparativa, muestra la diferencia de RMSE para la ejecución de k-NN sobre los diferentes regresores.

| Regresor | RollRate | PitchRate | CurrPitch | CurrRoll | DiffRollRate |
|---|---|---|---|---|---|
| **RMSE** | 0.0001942133 | 0.0003164413 | 0.0003065516 | 0.0002754811 | 0.0002598774 |

Por otra parte, el código R que muestra el gráfico con los ajustes del algoritmo para cada variable se muestra a continuación.

```
par(mfrow = c(2,3))
plot(Delta$RollRate,Delta$Sa, main = "KNN RollRate vs Sa", xlab =
"RollRate", ylab = "Sa")
points(Delta$RollRate, knn_fit_RollRate$fitted.values, col = "green",
pch=20)
plot(Delta$PitchRate,Delta$Sa, main = "KNN PitchRate vs Sa", xlab =
"PitchRate", ylab = "Sa")
points(Delta$PitchRate, knn_fit_PitchRate$fitted.values, col = "blue",
pch = 20)
plot(Delta$CurrPitch,Delta$Sa, main = "KNN CurrPitch vs Sa", xlab =
"CurrPitch", ylab = "Sa")
points(Delta$CurrPitch, knn_fit_CurrPitch$fitted.values, col = "red",
pch=20)
plot(Delta$CurrRoll,Delta$Sa, main = "KNN CurrRoll vs Sa", xlab =
"CurrRoll", ylab = "Sa")
points(Delta$CurrRoll, knn_fit_CurrRoll$fitted.values, col = "yellow",
pch=20)
plot(Delta$DiffRollRate, Delta$Sa, main = "KNN Diff vs Sa", xlab =
"DiffRollRate", ylab = "Sa")
points(Delta$DiffRollRate, knn_fit_DiffRollRate$fitted.values, col =
"brown", pch = 20)
```

A su vez, el gráfico que se obtiene como resultado de la ejecución del fragmento de código anterior puede apreciarse en la siguiente imagen.




**Modelos de regresión lineal multivariante**

En este otro apartado, se procederá a ejecutar el algoritmo k-NN sobre cada uno de los modelos de regresión multivariante realizados en la sección anterior de este trabajo. Del mismo modo que en el anterior epígrafe, solo se mostrará el código R de la ejecución del primer modelo de regresión lineal multivariante, al ser todos ellos análogos.

El código R que ejecuta el algoritmo k-NN para el modelo de regresión multivariante que aglutina todas las variables, es el siguiente:

```
knn_fit_all <- kknn(Delta$Sa ~., Delta, Delta)
yprime <- knn_fit_all$fitted.values
RMSE_fit_all <- sqrt(sum((Delta$Sa - yprime)^2)/length(yprime))
RMSE_fit_all
```

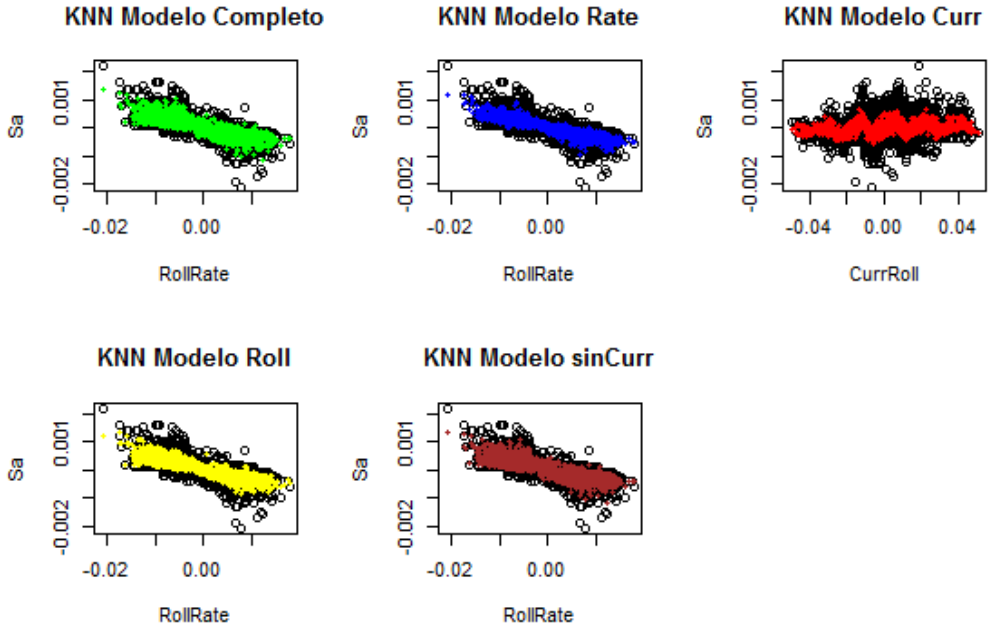
Análogamente a la sección de modelos de regresión lineal, la siguiente tabla muestra una comparativa del RMSE para cada uno de los modelos de regresión multivariante calculados.

| Regresor | fit_all | fit_rate | fit_curr | fit_roll | fit_sincurr |
|---|---|---|---|---|---|
| **RMSE** | 0.0001176665 | 0.0001543456 | 0.0002426328 | 0.0001437539 | 0.0001176665 |

Por otra parte, a continuación se muestra el código R que realiza el gráfico para visualizar el ajuste del algoritmo knn para cada uno de los modelos de regresión multivariante estudiados.

```
par(mfrow = c(2,3))
plot(Delta$RollRate, Delta$Sa, main = "KNN Modelo Completo", xlab =
"RollRate", ylab = "Sa")
points(Delta$RollRate, knn_fit_all$fitted.values, col = "green", pch =
20)
plot(Delta$RollRate, Delta$Sa, main = "KNN Modelo Rate", xlab =
"RollRate", ylab = "Sa")
points(Delta$RollRate, knn_fit_rate$fitted.values, col = "blue", pch =
20)
plot(Delta$CurrRoll, Delta$Sa, main = "KNN Modelo Curr", xlab =
"CurrRoll", ylab = "Sa")
points(Delta$CurrRoll, knn_fit_curr$fitted.values, col = "red", pch =
20)
plot(Delta$RollRate, Delta$Sa, main = "KNN Modelo Roll", xlab =
"RollRate", ylab = "Sa")
points(Delta$RollRate, knn_fit_Roll$fitted.values, col = "yellow", pch
= 20)
plot(Delta$RollRate, Delta$Sa, main = "KNN Modelo sinCurr", xlab =
"RollRate", ylab = "Sa")
points(Delta$RollRate, knn_fit_sincurr$fitted.values, col = "brown",
pch = 20)
```

Por su parte, el gráfico que se obtiene como resultado de la ejecución del código R anterior se muestra a continuación.






*Julio Alberto López Gómez*
*Juan Carlos Cubero, Coral del Val, Rocío Romero, Rafael Alcalá*

Visualmente, podemos comprobar cómo el ajuste de k-NN sobre los modelos de regresión multivariante parece mejor que el ajuste del algoritmo lm. Posteriormente, esta suposición realizada tras la visualización de los gráficos será demostrada y probada o refutada en la siguiente sección del trabajo.

**Modelos de regresión multivariante con interacciones y no linealidad**

Los últimos modelos sobre los que aplicar el algoritmo k-NN para estudiar su rendimiento, son los modelos de regresión multivariante con interacciones y no linealidad que se construyeron en la sección anterior. Para ello, el siguiente fragmento de código muestra el código R necesario para ejecutar el algoritmo k-NN sobre uno de estos modelos (del mismo modo que en los anteriores apartados, no se incluirá el código de la aplicación de k-NN sobre el resto de modelos por simplicidad y reducción del tamaño de la memoria).

```
knn_fit_int  <- kknn(Delta$Sa ~
Delta$RollRate*Delta$DiffRollRate*Delta$CurrRoll, Delta, Delta)
yprime <- knn_fit_int$fitted.values
RMSE_fit_int <- sqrt(sum((Delta$Sa - yprime)^2)/length(yprime))
RMSE_fit_int
```

A continuación, se muestra una tabla a modo de resumen que especifica el RMSE obtenido para cada uno de los modelos con interacciones y no linealidad estudiados en la sección anterior.

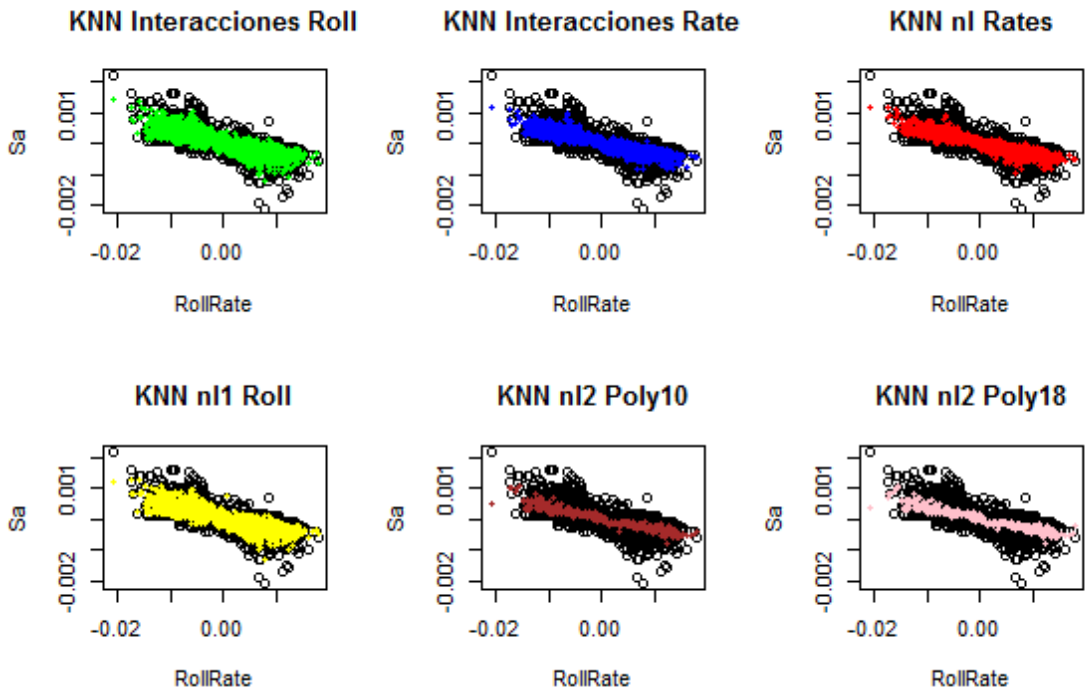
| Regresor | fit_int | fit_int2 | fit_nl | fit_nl1 | fit_nl2 | fit_nl3 |
|---|---|---|---|---|---|---|
| **RMSE** | 0.0001093035 | 0.0001472843 | 0.0001543456 | 0.0001223569 | 0.0001954295 | 0.0001950639 |

Una vez aplicado k-NN a todos los modelos con interacciones y no linealidad, se graficarán los resultados del ajuste de k-NN a cada uno de estos modelos. El código R desarrollado para tal efecto es el siguiente:

```
par(mfrow = c(2,3))
plot(Delta$RollRate, Delta$Sa, main = "KNN Interacciones Roll", xlab =
"RollRate", ylab= "Sa")
points(Delta$RollRate, knn_fit_int$fitted.values, col = "green", pch =
20)
plot(Delta$RollRate,Delta$Sa, main = "KNN Interacciones Rate", xlab =
"RollRate", ylab = "Sa")
points(Delta$RollRate, knn_fit_int2$fitted.values, col = "blue", pch =
20)
plot(Delta$RollRate, Delta$Sa, main = "KNN nl Rates", xlab =
"RollRate", ylab = "Sa")
```

```
points(Delta$RollRate, knn_fit_nl$fitted.values, col = "red", pch =
20)
plot(Delta$RollRate, Delta$Sa, main = "KNN nl1 Roll", xlab =
"RollRate", ylab = "Sa")
points(Delta$RollRate, knn_fit_nl1$fitted.values, col = "yellow", pch
= 20)
plot(Delta$RollRate, Delta$Sa, main = "KNN nl2 Poly10", xlab =
"RollRate", ylab = "Sa")
points(Delta$RollRate, knn_fit_nl2$fitted.values, col = "brown", pch =
20)
plot(Delta$RollRate, Delta$Sa, main = "KNN nl2 Poly18", xlab =
"RollRate", ylab = "Sa")
points(Delta$RollRate, knn_fit_nl3$fitted.values, col = "pink", pch =
20)
```

Y el resultado de ejecutar este código es el gráfico que aparece a continuación.




Por último, y para terminar el trabajo realizado en este apartado, mediremos la precisión del modelo utilizando el método k-fold cross-validation. Este método divide el conjunto de datos en k capas de tal forma que para cada k experimentos, elige una capa k como conjunto de entrenamiento y las restantes como conjunto de pruebas, por lo que obtiene muchos más resultados que permiten contrastar y evaluar la calidad del modelo al utilizarse todas las capas tanto como conjunto de entrenamiento como de pruebas. Los archivos de particiones para este dataset se han extraído del enlace http://sci2s.ugr.es/keel/dataset.php?cod=1256 .

Para ejecutar k-fold cross-validation, se ha seleccionado solo aquel modelo que dado su coeficiente de determinación $R^2$ y su RMSE resultaba ser el modelo más representativo para este conjunto de datos. Este modelo es aquel en el que se produce la interacción entre las variables Roll.

Una vez seleccionado el modelo, se construirán las funciones run_lm_fold y run_knn_fold que devolverán respectivamente, el MSE obtenido para los conjuntos de entrenamiento y test utilizando el algoritmo de regresión lineal (lm) y utilizando el algoritmo knn.

A continuación, se muestran, respectivamente, las funciones run_lm_fold y run_knn_fold y posteriormente, una tabla ilustrativa con los resultados obtenidos.

```
#################################################################
run_lm_fold <- function(i, x, tt = "test") {                   ##
  file <- paste(x, "-5-", i, "tra.dat", sep="");               ##
  x_tra <- read.csv(file, comment.char="@")                    ##
  file <- paste(x, "-5-", i, "tst.dat", sep="");               ##
  x_tst <- read.csv(file, comment.char="@")                    ##
  In <- length(names(x_tra)) - 1                               ##
  names(x_tra)[1:In] <- paste ("X", 1:In, sep="");             ##
  names(x_tra)[In+1] <- "Y"                                    ##
  names(x_tst)[1:In] <- paste ("X", 1:In, sep="");             ##
  names(x_tst)[In+1] <- "Y"                                    ##
  if (tt == "train") { test <- x_tra }                         ##
  else { test <- x_tst }                                       ##
  fitMulti=lm(Y~ X1*X5*X4,x_tra)                               ##
  yprime=predict(fitMulti,test)                                ##
  sum(abs(test$Y-yprime)^2)/length(yprime) ##MSE               ##
}                                                              ##
#################################################################
```

```
#################################################################
run_knn_fold <- function(i, x, tt = "test") {                  ##
  file <- paste(x, "-5-", i, "tra.dat", sep="")                ##
  x_tra <- read.csv(file, comment.char="@")                    ##
  file <- paste(x, "-5-", i, "tst.dat", sep="")                ##
  x_tst <- read.csv(file, comment.char="@")                    ##
  In <- length(names(x_tra)) - 1                               ##
  names(x_tra)[1:In] <- paste ("X", 1:In, sep="")              ##
  names(x_tra)[In+1] <- "Y"                                    ##
  names(x_tst)[1:In] <- paste ("X", 1:In, sep="")              ##
  names(x_tst)[In+1] <- "Y"                                    ##
  if (tt == "train") {                                         ##
    test <- x_tra                                              ##
  }                                                            ##
  else {                                                       ##
    test <- x_tst                                              ##
```



*Julio Alberto López Gómez*
*Juan Carlos Cubero, Coral del Val, Rocío Romero, Rafael Alcalá*

```
  }                                                          ##
  fitMulti=kknn(Y~ X1*X5*X4,x_tra,test)                      ##
  yprime=predict(fitMulti,test)                              ##
  sum(abs(test$Y-yprime)^2)/length(yprime) ##MSE             ##
}
```

Una vez mostradas las dos funciones, se adjunta el código R para calcular el MSE para lm y k-NN aplicado al modelo de interacciones utilizando k-fold cross-validation así como una tabla de resultados.

```
lmMSEtrain<-mean(sapply(1:5,run_lm_fold,nombre,"train"))
lmMSEtest<-mean(sapply(1:5,run_lm_fold,nombre,"test"))
knnMSEtrain<-mean(sapply(1:5,run_knn_fold,nombre,"train"))
knnMSEtest<-mean(sapply(1:5,run_knn_fold,nombre,"test"))
```

| Algoritmo | lm | Knn |
|---|---|---|
| **MSEtrain** | 2.942769e-08 | 1.201185e-08 |
| **MSEtest** | 2.959007e-08 | 3.040834e-08 |

Con los resultados provistos por la tabla anterior, podemos concluir que el Algoritmo lm presenta una variación del error menor que el algoritmo k-NN, que presenta una mayor diferencia entre el MSE cometido en el conjunto de entrenamiento y el MSE cometido en los conjuntos de prueba. Esto demuestra la afirmación de que no siempre un modelo que presenta un MSE muy pequeño en el conjunto de entrenamiento (como lo es el del algoritmo k-NN con respecto a lm) también presentará un MSE similar en los conjuntos de prueba, ya que como podemos comprobar el MSE en el conjunto de prueba es mucho mayor, mientras que el MSE presentado por lm para conjuntos de prueba es muy similar al que presentaba en los conjuntos de entrenamiento. Por este motivo, a priori, sin hacer las comparaciones que se llevarán a cabo en la siguiente sección, el algoritmo lm sería más adecuado que k-NN.

Si observamos los resultados de esta tabla y calculamos la raíz cuadrada del MSE, obteniendo el RMSE, podemos comprobar cómo los errores obtenidos tras aplicar k-fold cross-validation son en general levemente peores que los obtenidos anteriormente para los conjuntos de entrenamiento y mejores para los conjuntos de prueba. El código R que se ha utilizado para obtener estos datos es:

```
> RMSE_fit_all <- > RMSE_lm_fit_int    #RMSE para modelo con
interacciones
[1] 0.0001715911
> RMSE_fit_int        #RMSE para k-NN modelo con interacciones
[1] 0.0001093035
> sqrt(lmMSEtrain)   #RMSE para lm en train
[1] 0.000171545
> sqrt(lmMSEtest)    #RMSE para lm en test
```

```
[1] 0.0001720176
> sqrt(knnMSEtrain)  #RMSE para knn en train
[1] 0.0001095986
> sqrt(knnMSEtest)   #RMSE para knn en test
[1] 0.0001743799
```

**R-4. Comparar los resultados de los dos algoritmos de regresión múltiple entre sí, y adicionalmente mediante comparativas múltiples con un tercero (el modelo M5')**

En primer lugar, y antes de comenzar a realizar la comparativa entre los diferentes algoritmos, se van a editar los ficheros regr_test_alumnos y regr_train_alumnos con los datos hallados para lm y k-NN a lo largo del ejercicio. Estos datos son los correspondientes al modelo con interacciones que se ha analizado al final del anterior apartado.

Una vez que se han editado los ficheros, se cargarán éstos en dos tablas, que serán respectivamente tablatrain para el fichero que representa los datos obtenidos para diferentes datasets para los datos de entrenamiento y tablatst para los datos obtenidos en los conjuntos de prueba.

Una vez que han sido construidas las dos tablas, se aplicará el test estadístico de Wilcoxon. Para aplicar este test, es necesario normalizar el error, ya que los errores no se encuentran en el mismo rango, como sucede en los problemas de clasificación. Para ello, se utilizará la normalización propuesta en *M.J. Gacto, R. Alcalá, F. Herrera, Integration of an Index to Preserve the Semantic Interpretability in the Multi-Objective Evolutionary Rule Selection and Tuning of Linguistic Fuzzy Systems. IEEE Transactions on Fuzzy Systems 18:3 (2010) 515-531.* Una vez realizada esta normalización, se utilizará el código propuesto en la segunda sesión de laboratorio de regresión para aplicar los tests estadísticos de Wilcoxon y Friedman. Este código, puede verse en el fichero de la asignatura Statistical Linear Regression – Laboratorio2 y se adjunta a continuación.

```
#Tabla con los errores medios de entrenamiento
resultados <- read.csv("regr_train_alumnos.csv")
tablatra <- cbind(resultados[,2:dim(resultados)[2]])
colnames(tablatra) <- names(resultados)[2:dim(resultados)[2]]
rownames(tablatra) <- resultados[,1]
#Tabla con los errores medios de test
resultados <- read.csv("regr_test_alumnos.csv")
tablatst <- cbind(resultados[,2:dim(resultados)[2]])
colnames(tablatst) <- names(resultados)[2:dim(resultados)[2]]
rownames(tablatst) <- resultados[,1]


#Tabla difs que incluye la normalización propuesta
difs <- (tablatst[,1] - tablatst[,2]) / tablatst[,1]
```

```
wilc_1_2 <- cbind(ifelse (difs<0, abs(difs)+0.1, 0+0.1), ifelse
(difs>0, abs(difs)+0.1, 0+0.1))
colnames(wilc_1_2) <- c(colnames(tablatst)[1], colnames(tablatst)[2])
head(wilc_1_2)
#Aplicación del test de Wilcoxon
LMvsKNNtst <- wilcox.test(wilc_1_2[,1], wilc_1_2[,2], alternative =
"two.sided", paired=TRUE)
Rmas <- LMvsKNNtst$statistic
pvalue <- LMvsKNNtst$p.value
LMvsKNNtst <- wilcox.test(wilc_1_2[,2], wilc_1_2[,1], alternative =
"two.sided", paired=TRUE)
Rmenos <- LMvsKNNtst$statistic
Rmas
Rmenos
pvalue

#Aplicación del Test de Friedmann
test_friedman <- friedman.test(as.matrix(tablatst))
test_friedman
tam <- dim(tablatst)
groups <- rep(1:tam[2], each=tam[1])
pairwise.wilcox.test(as.matrix(tablatst), groups, p.adjust = "holm",
paired = TRUE)
```

Una vez aplicado el test de Wilcoxon, los resultados son los siguientes:

```
> Rmas
 V
50
> Rmenos
 V
41
> pvalue
[1] 0.7868652
```

Por lo que podemos concluir que no existen diferencias significativas entre ambos algoritmos, al existir únicamente un (1 - 0,78) = 22% de probabilidad de que existan diferencias entre LM y KNN.

Por otra parte, el test de Friedman permite hacer comparativas múltiples entre diferentes algoritmos, mientras que Wilcoxon compara únicamente dos. Los resultados de aplicar el test de Friedman a los datos obtenidos en este trabajo son:

```
Friedman rank sum test

data:  as.matrix(tablatst)
Friedman chi-squared = 7.3846, df = 2, p-value = 0.02491
```

Por lo que podemos concluir que sí que existen diferencias significativas entre al menos un par de algoritmos. Para estudiar y analizar entre qué algoritmos existen diferencias, se aplica el post-hoc Holm, cuyos resultados aparecen a continuación:

```
  1     2
2 0.588 -
3 0.098 0.115
```

En este caso concreto, existen diferencias significativas entre el algoritmo M5 y k-NN y entre M5 y LM a favor de M5 con aproximadamente un 90% de confianza, mientras que no se puede concluir que existan diferencias significativas entre LM y k-NN tal y como se vio en el test de Wilcoxon y cuya hipótesis respalda el test de Friedman.

## 3. Clasificación

**En este apartado el estudiante debe utilizar el dataset$_c$ asignado, en este caso Heart, para realizar lo siguiente:**

**C-1. Utilizar el algoritmo k-NN probando con diferentes valores de k. Elegir el que considere más adecuado para su conjunto de datos.**

Para realizar este apartado, se utilizará la función para ejecutar el algoritmo k-NN presente en el paquete caret, que tendrá que ser cargado previamente mediante la función require(caret)

```
require(caret)
```

Una vez cargado este paquete, se crearán los conjuntos de entrenamiento y pruebas para los datos de Heart. Para ello, se utilizará el 70% del conjunto como conjunto de entrenamiento y el 30% restante como conjunto de pruebas. Ambos conjuntos serán aleatorios. Esto se consigue mediante la ejecución del siguiente código R.

```
#Se crean las particiones de los datos (70% Entrenamiento y 30%
Pruebas)
trainIndex <- createDataPartition(Heart$Class, p = .7, list = FALSE,
times = 1)
#Se crea el conjunto de entrenamiento sin la clase y se obtiene la
clase aparte
Heart_train <- Heart[trainIndex, c(-length(Heart))]
Heart_train_labels <- as.factor(Heart[trainIndex, ]$Class)
#Se crea el conjunto de prueba sin la clase y se obtiene la clase
aparte
Heart_test <- Heart[-trainIndex, c(-length(Heart))]
Heart_test_labels <- as.factor(Heart[-trainIndex, ]$Class)
```

Con los datos cargados y preparados para ejecutar el algoritmo, se ejecutará k-NN con todos los K's posibles. Esta ejecución es larga y costosa, pero ayudará a determinar cuál es el K óptimo y precisar después esta diferencia con respecto al resto de K. Este conjunto de operaciones, se reflejan en el siguiente código R.

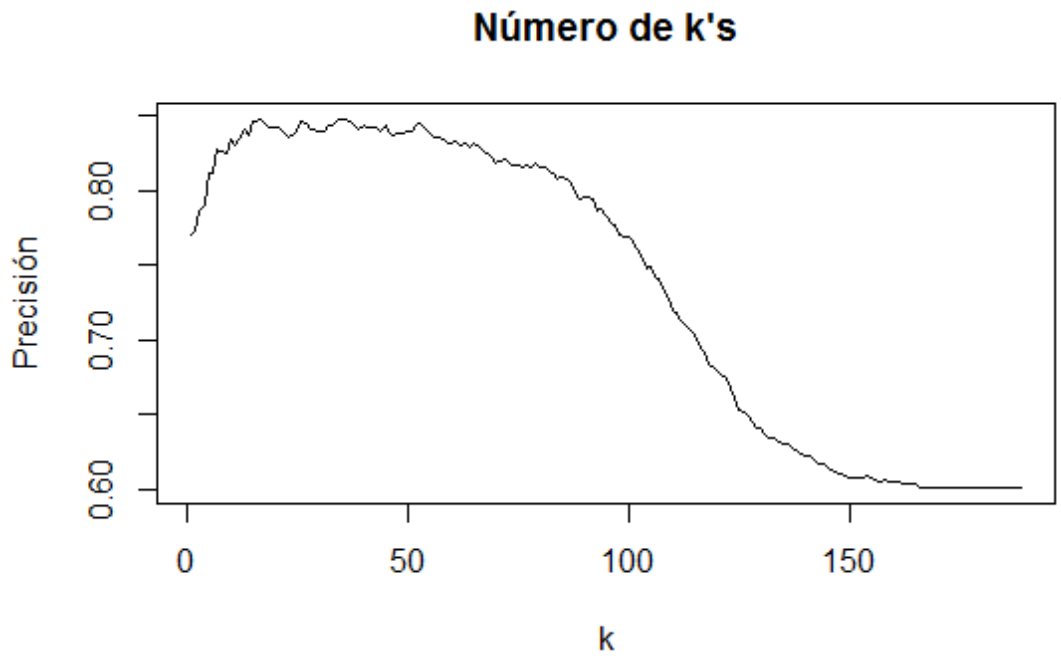
```
knnFit <- train(Heart_train, Heart_train_labels, method="knn",
metric="Accuracy", preProc = c("center", "scale"),
                tuneGrid = data.frame(.k=1:dim(Heart_train)[1]))
knnPred <- predict(knnFit, newdata = Heart_test)
postResample(pred = knnPred, obs = Heart_test_labels)
```

Donde como se puede ver, se ejecuta la función de entrenamiento con el conjunto de entrenamiento y las clases para dichos elementos, utilizando el algoritmo k-NN. Posteriormente, se ejecuta el algoritmo con el conjunto de prueba y se utiliza la función postResample() para medir la precisión del algoritmo, obteniendo un 86,2% de eficacia.

Si observamos el valor de la variable knnFit, podemos ver el resultado del algoritmo para cada uno de los K elegidos y finalmente, el mejor valor obtenido por el algoritmo, que es K = 35.

Una vez ejecutada y medida la precisión del algoritmo, es posible graficar la precisión de este con respecto al número de vecinos elegido. Para ello, se ejecuta el siguiente código R, que da como resultado el gráfico que aparece a continuación.

```
plot(knnFit$results[,2], type="l", xlab="k", ylab = "Precisión",
main="Número de k's")
```

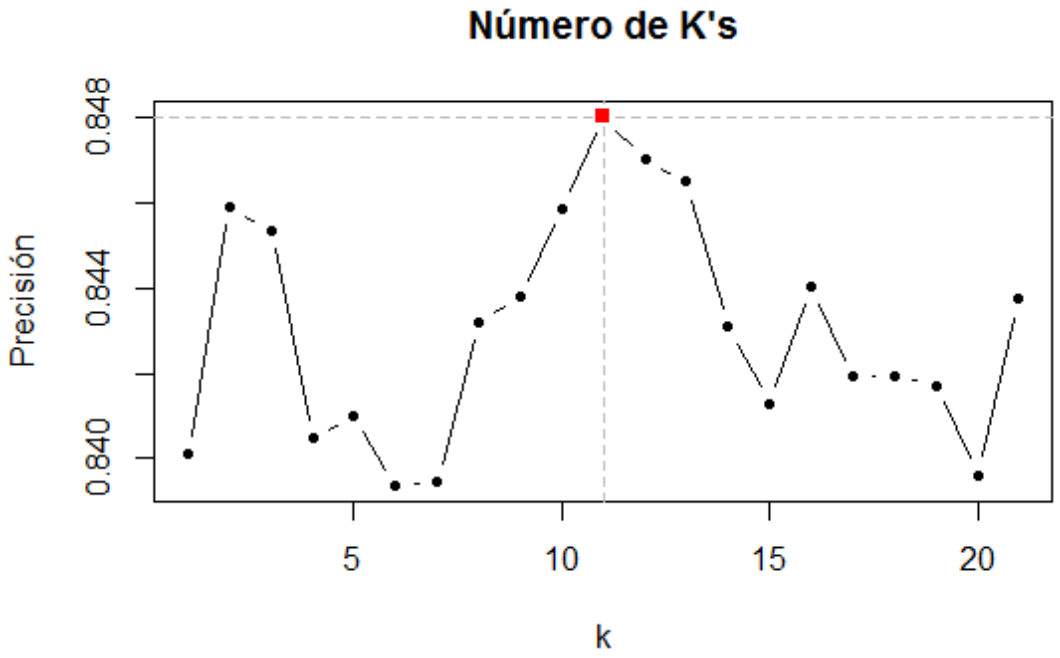



En el gráfico anterior, es posible apreciar cómo a medida que aumenta el número de k, disminuye la precisión del modelo. Del mismo modo, cuando el número de k es muy pequeño, los resultados tampoco son buenos. Esto se conoce como el problema de la negociación bias-variance.

Una vez ejecutado el algoritmo y sabiendo que K = 35 es el número óptimo de vecinos, se dibujará la gráfica anterior en un entorno de diez K's por encima y por debajo del K óptimo para apreciar de manera más exacta el óptimo obtenido por el algoritmo.

Para ello, será necesario ejecutar el siguiente código R, que producirá el gráfico que se muestra a continuación

```
ub <- knnFit[[6]][1,1]-10
lwb <- knnFit[[6]][1,1]+10
plot(knnFit$results[lwb:ub,1],knnFit$results[lwb:ub,2], type="b",
xlab="k", ylab = "Precisión", main="Número de K's", pch = 20)
abline(h = knnFit$results[knnFit[[6]][1,1],2], col = "gray", lty =
"dashed")
abline(v = knnFit$results[knnFit[[6]][1,1],1], col = "gray", lty =
"dashed")
points(knnFit[[6]][1,1], knnFit$results[knnFit[[6]][1,1],2], col =
"red", pch=15)
```




En el código anterior, se muestra cómo se dibujan los resultados de k-NN en un entorno de 10 K por encima y por debajo del K óptimo. Para ello, en la estructura que devuelve la ejecución de k-NN se ha capturado el valor del K óptimo, que se corresponde con el sexto elemento de la lista que devuelve la ejecución de la función k-NN y se grafican los valores de diez K por encima y por debajo del K óptimo obtenido.

**C-2. Utilizar el algoritmo LDA para clasificar.**

En este apartado, en lugar de utilizar el algoritmo k-NN para clasificar, se utilizará el algoritmo LDA. Para ello, el código R ejecutado, se muestra a continuación.

```
ldafit <- lda(Heart$Class ~ Heart$Age + Heart$Sex +
Heart$ChestPainType + Heart$RestBloodPressure +
Heart$SerumCholestoral + Heart$FastingBloodSugar +
Heart$ResElectrocardio + Heart$MaxHeartRate + Heart$OldPeak +
Heart$Slope + Heart$MajorVessels + Heart$Thal, data=Heart,
subset=trainIndex)
ldaPred_train <- predict(ldafit, new_data = Heart_train_labels)
ldafit <- lda(Heart$Class ~ Heart$Age + Heart$Sex +
Heart$ChestPainType + Heart$RestBloodPressure +
Heart$SerumCholestoral + Heart$FastingBloodSugar +
Heart$ResElectrocardio + Heart$MaxHeartRate + Heart$OldPeak +
Heart$Slope + Heart$MajorVessels + Heart$Thal, data=Heart, subset=-
trainIndex)
ldaPred_test <- predict(ldafit, new_data = Heart_test_labels)
ldacm_train <- confusionMatrix(data = ldaPred_train$class, reference =
Heart_train_labels)
ldacm_test <- confusionMatrix(data = ldaPred_test$class, reference =
Heart_test_labels)
ldacm_train
ldacm_test
confusion_matrix <- ldacm_train[[2]] + ldacm_test[[2]]
confusion_matrix
```

Como se puede observar atendiendo al código adjunto, se usarán las particiones empleadas para la ejecución del algoritmo k-NN, de manera que en el apartado C-4 será posible comparar estos algoritmos al haberse ejecutado en idénticas condiciones.

Por otra parte, con la función lda se entrena el modelo. Cabe destacar que este modelo está formado por todas las variables a excepción de ExerciseInduced, ya que esta variable contenía valores nulos y la función no podía entrenar dicho modelo. Esta operación se realiza dos veces, una entrenando el modelo con el conjunto de entrenamiento y otra donde el entrenamiento se realiza con el conjunto de prueba. De esta forma, se realizan las predicciones con los respectivos conjuntos y se calculan las matrices de confusión en ambos casos para ver la efectividad del clasificador sobre el conjunto de entrenamiento y sobre el conjunto de prueba.

Finalmente, y para resumir los datos obtenidos, se han unido en una única matriz de confusión llamada confusion_matrix_lda las dos matrices de confusión creadas anteriormente, obteniendo así la siguiente matriz de confusión.

```
          Reference
Prediction   1   2
         1 137  23
         2  13  96
```

A partir de esta matriz, se puede observar cómo el algoritmo clasifica bastante bien.Este hecho viene a confirmar lo que se vio a la hora de crear las dos matrices de confusión, y es que existía un 86% de acierto en la clasificación para ambos modelos.

**C-3. Utilizar el algoritmo QDA para clasificar.**

De forma análoga a la aplicación del algoritmo LDA para el conjunto de datos Heart, en esta sección se aplicará el algoritmo QDA para este mismo conjunto de datos. La aplicación del algoritmo QDA es prácticamente idéntica a la realizada en la sección anterior para LDA.

En este caso, también se trabajará con las mismas particiones que se hicieron al principio del apartado de clasificación, con el objetivo de aplicar todos los algoritmos bajo las mismas condiciones para, posteriormente, realizar la comparativa que se pide en la sección C-4.

Posteriormente, se entrenará el modelo QDA con los datos del conjunto de entrenamiento, se realizarán las predicciones sobre ese conjunto y se creará la matriz de confusión para estudiar la efectividad del clasificador realizado con este algoritmo para los datos de entrenamiento. Posteriormente, se realizarán las mismas operaciones que se acaban de describir, solo que con los datos del conjunto de prueba, entrenando primero el modelo con dichos datos, realizando las predicciones y calculando su matriz de confusión para ver la efectividad del clasificador para los datos del conjunto de prueba. Para resumir estos datos, las dos matrices de confusión se sumarán en la matriz confusion_matrix_qda.

El código R que realiza las operaciones descritas arriba es el siguiente:

```
qdafit <- qda(Heart$Class ~ Heart$Age + Heart$Sex +
Heart$ChestPainType + Heart$RestBloodPressure +
Heart$SerumCholestoral + Heart$FastingBloodSugar +
Heart$ResElectrocardio + Heart$MaxHeartRate + Heart$OldPeak +
Heart$Slope + Heart$MajorVessels + Heart$Thal, data=Heart,
subset=trainIndex)
qdaPred_train <- predict(qdafit, new_data = Heart_train_labels)
qdacm_train <- confusionMatrix(data = qdaPred_train$class, reference =
Heart_train_labels)
qdacm_train
qdafit <- qda(Heart$Class ~ Heart$Age + Heart$Sex +
Heart$ChestPainType + Heart$RestBloodPressure +
```

```
Heart$SerumCholestoral + Heart$FastingBloodSugar +
Heart$ResElectrocardio + Heart$MaxHeartRate + Heart$OldPeak +
Heart$Slope + Heart$MajorVessels + Heart$Thal, data=Heart, subset=-
trainIndex)
qdaPred_test <- predict(qdafit, new_data = Heart_test_labels)
qdacm_test <- confusionMatrix(data = qdaPred_test$class, reference =
Heart_test_labels)
qdacm_test
confusion_matrix_qda <- qdacm_train[[2]] + qdacm_test[[2]]
confusion_matrix_qda
```

Donde la matriz de confusión confusion_matrix_qda es la siguiente:

```
          Reference
Prediction   1   2
         1 138  18
         2  12 101
```

Y los porcentajes de precisión obtenidos han sido: un 88% de precisión para el modelo entrenado con el conjunto de entrenamiento y un 90% para el entrenado con el conjunto de prueba.

### C-4. Comparar los resultados de los tres algoritmos

En primer lugar, para realizar la comparativa de los tres métodos propuestos en este apartado de clasificación, se utilizará el test de Wilcoxon para comparar los algoritmos LDA y QDA. Posteriormente, se utilizará el test de Friedman para hacer comparativas múltiples entre los tres algoritmos vistos en este apartado: k-NN, LDA y QDA.

De este modo, en primer lugar, se cargarán los datos de entrenamiento para el conjunto de Datos Heart, y se aplicará el test de Wilcoxon para los valores obtenidos por LDA y QDA, lo cual se refleja en el siguiente código adjunto:

```
train <- read.csv("clasif_train_alumnos.csv", header=T, sep= ",")
wt <- wilcox.test(train$out_train_lda, train$out_train_qda,
alternative = "two.sided", paired=TRUE)
wt
```

Donde los resultados de este test, son:

```
        Wilcoxon signed rank test

data:  train$out_train_lda and train$out_train_qda
V = 68, p-value = 0.1769
alternative hypothesis: true location shift is not equal to 0
```

Siendo el p-valor = 0,1769 por lo que podemos concluir que no existen diferencias significativas entre estos dos algoritmos.

A continuación, se aplicará el test de Friedman para comparativas múltiples entre k-NN, LDA y QDA, ejecución que viene dada por el siguiente fragmento de código.

```
x <- as.matrix(train[,2:4])
fm <- friedman.test(x)
fm
```

Los resultados de este test son:

```
Friedman rank sum test

data:  x
Friedman chi-squared = 1.3, df = 2, p-value = 0.522
```

Donde el p-valor es igual a 0,522 y por tanto podemos concluir que no existen diferencias significativas entre al menos un par de algoritmos.

Con fines principalmente didácticos, a continuación se estudiará cómo comprobar entre qué algoritmos existen diferencias y a favor de quién, en caso de que las hubiera. Para ello, se aplicará post-hoc Holm. Para ello, se ejecutará el siguiente fragmento de código.

```
tam <- dim(x)
groups <- rep(1:tam[2], each=tam[1])
pairwise.wilcox.test(x, groups, p.adjust = "holm", paired = TRUE)
```

Los resultados de este método, se muestran a continuación.

```
  1    2
2 0.66 -
3 0.66 0.53
```

Lo que refuerza la hipótesis obtenida por los test de Wilcoxon y Friedman, donde el pri mero concluía que no existen diferencias significativas entre LDA y QDA, mientras que el segundo concluye que no existen diferencias significativas entre los tres algoritmos, hecho que se puede comprobar con los resultados dados por el post-hoc Holm.

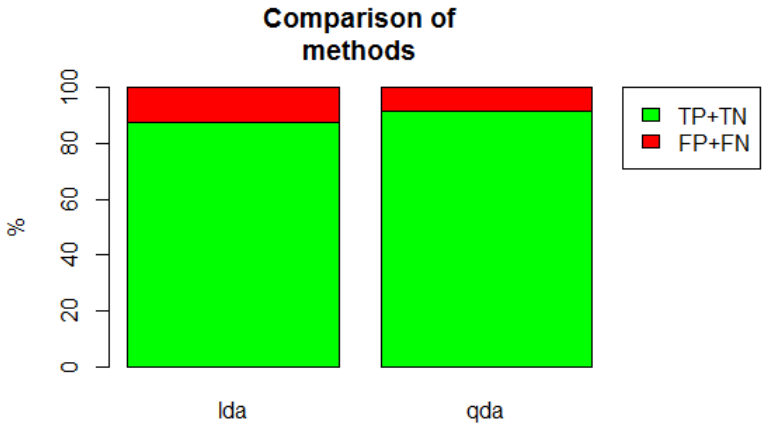
Finalmente, y para apoyar los resultados ofrecidos por los test estadísticos, se realizarán un par de gráficos que, en caso de haber aplicado los test correctamente, corroborarán las conclusiones obtenidas de ellos.

1. Gráfico comparativo de los algoritmos LDA y QDA

Este primer gráfico, se trata de un diagrama de barras apiladas que ofrece una comparativa entre los aciertos y fallos para LDA y QDA. Para ello, se utilizará el código propuesto por Rocío Romero Zaliz en el fichero Ejercicios_Comentarios_2015_2016.pdf, donde a partir de las matrices de confusión de ambos algoritmos, se extraen los aciertos obtenidos por éstos así como sus fallos mediante las funciones definidas aciertos(x) y fallos(x). Únicamente se ha modificado en este código los parámetros de inset para que la leyenda no se saliera del área del gráfico.

El código R propuesto por Rocío Romero así como el gráfico generado, se muestran a continuación.

```
aciertos <- function(x) {
  (x[1,1] + x[2,2]) / sum(x)
}
errores <- function(x) {
  (x[1,2] + x[2,1]) / sum(x)
}
tabla_res <- data.frame(lda=c(aciertos(confusion_matrix_lda),
errores(confusion_matrix_lda)), qda=c(aciertos(confusion_matrix_qda),
errores(confusion_matrix_qda)))
par(mar = c(5.1, 4.1, 4.1, 7))
par(xpd=TRUE)
barplot(as.matrix(tabla_res)* 100, ylab = "%", col = c("green",
"red"), main="Comparison of
        methods")
legend("topright", inset=c(-0.3,0), fill=c("green","red"), legend =
c("TP+TN", "FP+FN"))
par(xpd=FALSE)
par(mar = c(5.1, 4.1, 4.1, 2.1))
```






*Julio Alberto López Gómez*
*Juan Carlos Cubero, Coral del Val, Rocío Romero, Rafael Alcalá*

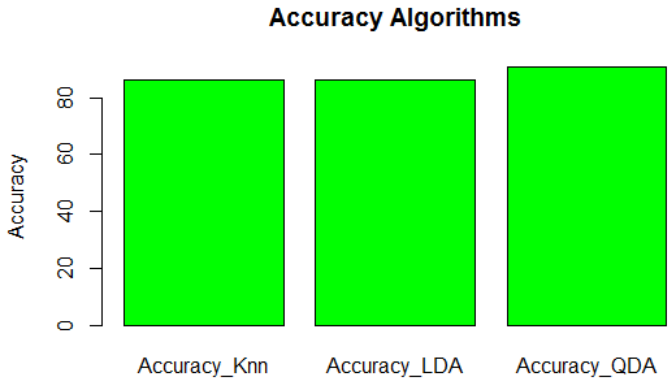
Donde tal y como se puede comprobar, aunque el algoritmo QDA es más eficaz que LDA, ambos algoritmos son muy similares, tal y como concluyeron los test estadísticos realizados.

2. Gráfico comparativo de precisión en la predicción para cada algoritmo

Para la realización de este gráfico, se han utilizado los porcentajes de precisión obtenidos por k-NN, LDA y QDA, construyendo con ellos una matriz que ha sido representada en un diagrama de barras. El código R que obtiene estos valores, construye la matriz y genera el gráfico, puede verse a continuación.

```
Accuracy_Knn <- postResample(pred = knnPred_train, obs =
Heart_train_labels)
Accuracy_Knn <- Accuracy_Knn[[1]]
Accuracy_Knn
Accuracy_LDA <- aciertos(confusion_matrix_lda)
Accuracy_LDA
Accuracy_QDA <- aciertos(confusion_matrix_qda)
Accuracy_QDA
Precision <- cbind(Accuracy_Knn, Accuracy_LDA, Accuracy_QDA)
barplot(as.matrix(Precision)*100, main = "Accuracy Algorithms" ,ylab =
"Accuracy", col=(c("green")))
```

A su vez, el gráfico generado se muestra a posteriori:




Donde como se puede comprobar, la eficacia del algoritmo QDA es mayor con respecto a k-NN y LDA, aunque tal y como concluyeron los test estadísticos, no existen diferencias significativas entre ninguno de ellos, habiendo obtenidos todos ellos un porcentaje de éxito en la clasificación bastante aceptable.