Tipología y ciclo de vida de los datos: PRA2 - Limpieza y análisis de datos

Autor: Giovanny Caluña e Ivan Ovalle

Mayo 2021

Contents

Detalles de la actividad

Descripción

En esta práctica se elabora un caso práctico orientado a aprender a identificar los datos relevantes para un proyecto analítico y usar las herramientas de integración, limpieza, validación y análisis de las mismas.

Objetivos

Los objetivos que se persiguen mediante el desarrollo de esta actividad práctica son los siguientes: - Aprender a aplicar los conocimientos adquiridos y su capacidad de resolución de problemas en entornos nuevos o poco conocidos dentro de contextos más amplios o multidisciplinares. - Saber identificar los datos relevantes y los tratamientos necesarios (integración, limpieza y validación) para llevar a cabo un proyecto analítico. Aprender a analizar los datos adecuadamente para abordar la información contenida en los datos. - Identificar la mejor representación de los resultados para aportar conclusiones sobre el problema planteado en el proceso analítico. - Actuar con los principios éticos y legales relacionados con la manipulación de datos en función del ámbito de aplicación. - Desarrollar las habilidades de aprendizaje que permita continuar estudiando de un modo que tendrá que ser en gran medida autodirigido o autónomo. - Desarrollar la capacidad de búsqueda, gestión y uso de información y recursos en el ámbito de la ciencia de datos.

Competencias

Las competencias que se desarrollan son:

- Capacidad de analizar un problema en el nivel de abstracción adecuado a cada situación y aplicar las habilidades y conocimientos adquiridos para abordarlo y resolverlo.
- Capacidad para aplicar las técnicas específicas de tratamiento de datos (integración, transformación, limpieza y validación) para su posterior análisis.

```
# Se cargan las librerías a usar:
library(ggplot2)
library(ggpubr)
```

```
## Warning: package 'ggpubr' was built under R version 4.0.5
```

```
library(grid)
library(gridExtra)
library(C50)
```

```
## Warning: package 'C50' was built under R version 4.0.5
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:gridExtra':
##
##     combine

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v tibble  3.1.0     v purrr   0.3.4
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## Warning: package 'readr' was built under R version 4.0.5

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::combine() masks gridExtra::combine()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
```

```
library(nortest)
library(ResourceSelection)
```

```
## Warning: package 'ResourceSelection' was built under R version 4.0.5

## ResourceSelection 0.3-5   2019-07-22
```

```
library(pROC)
```

```
## Warning: package 'pROC' was built under R version 4.0.5

## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.5
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
#install.packages('e1071', dependencies=TRUE)
library(rpart)
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 4.0.5
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.0.5
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:gridExtra':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(tictoc)
```

```
## Warning: package 'tictoc' was built under R version 4.0.5
```

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

1.1 Descripción del dataset

Para el desarrollo de este proyecto se seleccionó el data set llamado: Titanic: Machine Learning. Obtenido de: (https://www.kaggle.com/c/titanic). El data set cuenta con la información de diferentes tripulantes que estuvieron en el Titanic el dia de su hundimineto.

Lectura del fichero

Para cargar los datos en un data frame compatible con nuestro entorno, leemos el fichero de tipo .csv, utilizando la función *read.csv* de R.

```
passengers<-read.csv("./train.csv",header=T,sep=",")
```

El data set cuenta con 12 atributos que ayudan a describir a cada pasajero:

- PassengerID: Id del pasajero registrado.
- Survived: Si el pasajero sobrevivió o no al incidente(0 = No, 1 = Si).
- Pclass: Clase del ticket abordo del Titanic. Ej: 1,2 o 3.
- Name: Nombre del pasajero.
- Sex: Genero del pasajero.
- Age: Edad del pasajero.
- SibSp: Número de hermanos o conyuges a bordo del Titanic.
- Parch: Número de padres o niños a bordo del Titanic.
- Ticket: Número del ticket.
- Fare: Tarifa del pasajero (costo del ticket).
- Cabin: Número de cabina abordo del titanic.
- Embarked: Puerto de embarcación (C = Cherbourg, Q = Queenstown, S = Southampton)

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

Tomando en cuenta la información que nos provee el data set, el siguiente trabajo desarrollará tres modelos supervisados (Árbol de Decisión, Regresión Logística y Rain Forest) para predecir si un pasajero sobrevivió o no al hundimiento del Titanic, basado en los diferentes atributos registrados para cada pasajero. Además, se comparará la precisión de los diferentes modelos. Por último, los modelos entrenados nos ayudarán a identificar los atributos que más influyeron en la supervivencia o no de un pasajero.

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

Ahora utilizaremos la funcion *sapply*, que nos proporcionará el tipo de dato que maneja cada atributo del data set.

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

```
## PassengerId    Survived      Pclass        Name         Sex         Age
##   "integer"   "integer"   "integer" "character" "character"   "numeric"
##       SibSp       Parch      Ticket        Fare       Cabin    Embarked
##   "integer"   "integer" "character"   "numeric" "character" "character"
```

Como podemos observar en la tabla superior, en nuestro data set podemos encontrar 3 tipos de datos: integer, character y numeric. Ahora procedemos a utilizar la funcion *summary* que nos ayudará con datos estadisticos generales de cada atributo.

```
summary(passengers)
```

```
##   PassengerId       Survived          Pclass          Name
##  Min.   :  1.0   Min.   :0.0000   Min.   :1.000   Length:891
##  1st Qu.:223.5   1st Qu.:0.0000   1st Qu.:2.000   Class :character
##  Median :446.0   Median :0.0000   Median :3.000   Mode  :character
##  Mean   :446.0   Mean   :0.3838   Mean   :2.309
##  3rd Qu.:668.5   3rd Qu.:1.0000   3rd Qu.:3.000
##  Max.   :891.0   Max.   :1.0000   Max.   :3.000
##
##      Sex                 Age            SibSp           Parch
##  Length:891         Min.   : 0.42   Min.   :0.000   Min.   :0.0000
##  Class :character   1st Qu.:20.12   1st Qu.:0.000   1st Qu.:0.0000
##  Mode  :character   Median :28.00   Median :0.000   Median :0.0000
##                     Mean   :29.70   Mean   :0.523   Mean   :0.3816
##                     3rd Qu.:38.00   3rd Qu.:1.000   3rd Qu.:0.0000
##                     Max.   :80.00   Max.   :8.000   Max.   :6.0000
##                     NA's   :177
##     Ticket               Fare           Cabin             Embarked
##  Length:891         Min.   :  0.00   Length:891         Length:891
##  Class :character   1st Qu.:  7.91   Class :character   Class :character
##  Mode  :character   Median : 14.45   Mode  :character   Mode  :character
##                     Mean   : 32.20
##                     3rd Qu.: 31.00
##                     Max.   :512.33
##
```

Se obtiene el mínimo, la media y la mediana de todos los atributos de tipo integer y numeric. Además, en el atributo Age (edad del pasajero) podemos observar un campo extra llamado NA´s. Este dato nos indica el numero de valores nulos o vacios que contienes este campo, en este caso 177.

Para la eliminación de atributos innecesarios utilizaremos la funcion *str* de R que nos mostrará algunas muestras de cada atributo asi como su formato.

```
str(passengers)
```

```
## 'data.frame':    891 obs. of  12 variables:
##  $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Survived   : int  0 1 1 1 0 0 0 0 1 1 ...
##  $ Pclass     : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ Name       : chr  "Braund, Mr. Owen Harris" "Cumings, Mrs. John Bradley (Florence Briggs Thayer)"
##  $ Sex        : chr  "male" "female" "female" "female" ...
##  $ Age        : num  22 38 26 35 35 NA 54 2 27 14 ...
##  $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket     : chr  "A/5 21171" "PC 17599" "STON/O2. 3101282" "113803" ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : chr  "" "C85" "" "C123" ...
##  $ Embarked   : chr  "S" "C" "S" "S" ...
```

El primer atributo que se eliminará es: Passenger Id. Este atributo nos indica un número asignado de manera incremetal a cada pasajero por lo que no nos aporta información relevante para el análisis y entrenamiento del modelo. El segundo atributo eliminado será Name, este atributo de tipo character al ser diferente para cada pasajero, no proporciona informacion útil para la clasificación. De la misma manera, el atributo ticket, da un valor alfanumerico especifico para cada pasajero por lo que tambien se eliminará. Por ultimo, después de un análisis visual sobre el data set, se detecto un alto número de valores nulos sobre el campo Cabin y los pocos registros con los que se cuenta dan un valor específico por pasajero. Para eliminar los atributos mecionados, se procede a ejecutar lo siguiente:

```
passengers$PassengerId<- NULL
passengers$Name <- NULL
passengers$Ticket <- NULL
passengers$Cabin <- NULL
```

3. Limpieza de los datos

El primer paso para elaborar nuestro modelo de clasificaion es: la limpieza de los datos. En esta etapa vamos a aplicar las diferentes técnicas de limpieza de datos que nos perimitirá corregir posibles inconsistencias, valores nulos y atributos innecesarios.

3.1 Valores nulos

Ahora procederemos a tratar los valores nulos. Gracias a una inspeccion "manual" de los datos, se a conseguido identificar valores nulos en el campo Age de tipo NA y valores nulos en el campo Fare de tipo "0".

Aplicamos el siguiente comando que nos ayudara a contabilizar los valores nulos en cada atributo:

```
#Encuentra valos NA
colSums(is.na(passengers))
```

```
## Survived   Pclass      Sex      Age    SibSp    Parch     Fare Embarked
##        0        0        0      177        0        0        0        0
```

```
#Encuentra valores de tipo numeric igual a 0
colSums(passengers == 0)
```

```
## Survived   Pclass      Sex      Age    SibSp    Parch     Fare Embarked
##      549        0        0       NA      608      678       15        0
```

Con la funcion *is.na* podmeos observar que el data set cuenta con varios datos vacios o nulos en el atributo Age (edad). Debido al alto numero de valores nulos y a su alta impacto en el analisis, vamos a revisar de manera detenida el atributo.

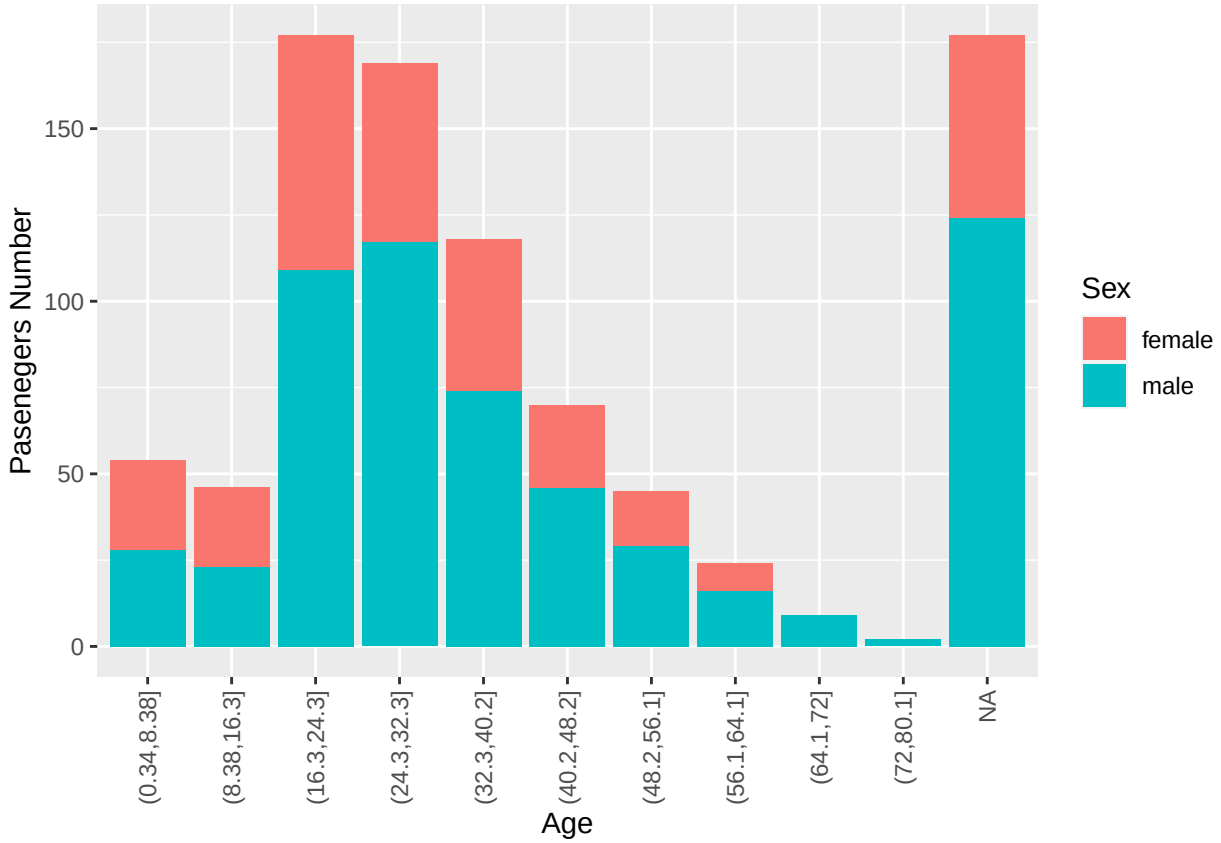
Con el siguiente comando, vamos a ordenar los valores que toma el atributo basado en el número de apariciones en el data set.

```
sort(table(passengers$Age), decreasing = TRUE)
```

```
##
##    24    22    18    19    28    30    21    25    36    29    26    27    32    35    16    31
##    30    27    26    25    25    25    24    23    22    20    18    18    18    18    17    17
##    20    23    33    34    39    17    40    42    45    38     2     4    50    44    47    48
##    15    15    15    15    14    13    13    13    12    11    10    10    10     9     9     9
##     9    54     1    51     3    14    37    41    49    52    15    43    58     5     8    11
##     8     8     7     7     6     6     6     6     6     6     5     5     5     4     4     4
##    56    60    62     6     7    46    61    65 0.75 0.83    10    13 28.5 30.5 32.5 40.5
##     4     4     4     3     3     3     3     3     2     2     2     2     2     2     2     2
## 45.5    55    57    59    63    64    70    71 0.42 0.67 0.92    12 14.5 20.5 23.5 24.5
##     2     2     2     2     2     2     2     2     1     1     1     1     1     1     1     1
## 34.5 36.5    53 55.5    66 70.5    74    80
##     1     1     1     1     1     1     1     1
```

La primera fila representa los valores con mayor frecuencia, aqui se encuentran valores en el rango de 20 a 30 años aproximadamente. Para poder visualizar la informacion de manera grafica, cargamos las librearias necesarias y procedemos a plotear la información con el siguiente comando:

```
ggplot(passengers, aes(cut(Age,10), fill = Sex)) + geom_bar()+ guides(x = guide_axis(angle = 90))+labs(
```

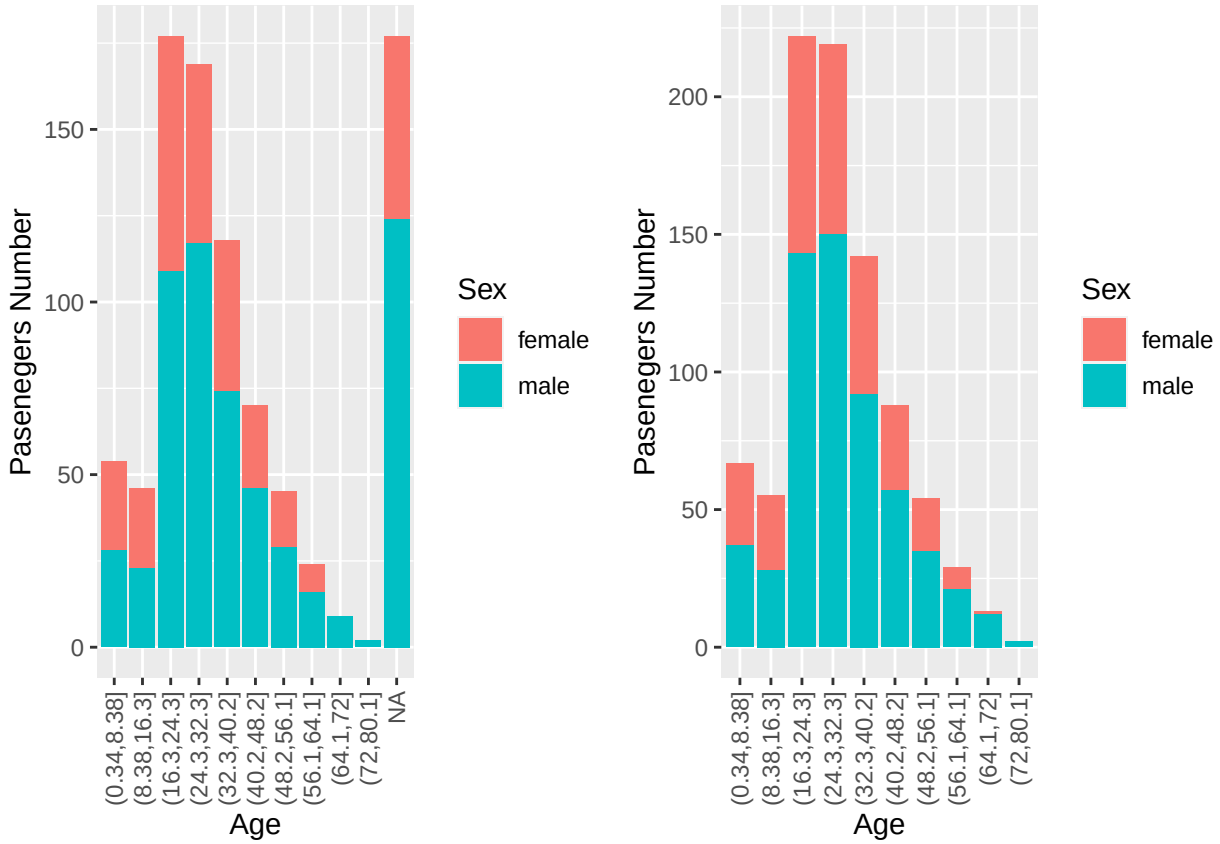


Como se puede observar en el gráfico, la mayor parte de pasajeros se encuentra distribuido en edades que van desde los 16.3 años hasta los 32.3. Ademas, podemos apreciar que el numero de hombres y mujeres mantienen una relacion de proporcion en todos los intervalos. Por esta razon, se procede a llenar los datos faltantes del atibuto edad con valores aleatorios obtenidos del mismo data set, sin distincion si es hombre o mujer. Para esto se ejecuta el siguiente comando:

```
#Se fija una semilla para que el trabajo se pueda repetir sin alterar los resultados
set.seed(873465)
#Guardamos el atributo edad para comparar la distribucion una vez que los valores nulos sean reemplazado
tmpAge <- passengers$Age
#Creamos una variable que contenga todas las edades de los pasajeros
tmp <- passengers$Age[!is.na(passengers$Age)]
#Reemplazamos los valores nulos con un valor aleatorio de nuestra variable tmp.
passengers$Age[is.na(passengers$Age)]  <- sample(tmp, size = 177, replace = FALSE)
```

Una vez reemplazados los valores procedemos a graficarlos para comparar los resultados y verificar si hubo alguna alteracion significativa en la distribucion de datos.

```
plotAfter <- ggplot(passengers, aes(cut(Age,10), fill = Sex)) + geom_bar()+ guides(x = guide_axis(angle
plotBefore <- ggplot(passengers, aes(cut(tmpAge,10), fill = Sex)) + geom_bar()+ guides(x = guide_axis(a
grid.arrange(plotBefore,plotAfter,ncol=2)
```



Como se puede observar en el grafico derecho, despues de la asignacion de valores, ya no existen valores nulos. Ademas, a pesar del alto numero de valores nulos con el metodo aplicado la diferencia en la distribucion de edades es minima.

Ahora procedemos a analizar el atributo fare. Como se calculó en el apartado anterior, existen 15 valores del atributo fare que son 0. Para reemplazar este valor, vamos a calcular la mediana de los demas registros.

```
tmpFare <- passengers$Fare[passengers$Fare !=0]
summary(tmpFare)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.013   7.925  14.500  32.756  31.275 512.329
```

```
passengers$Fare[passengers$Fare==0]  <- median(tmpFare)
summary(passengers$Fare)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.013   7.925  14.500  32.448  31.000 512.329
```

Como se puede observar, el reemplazo de valores apenas afecta la media global del data set

3.2 Normalizacion de datos

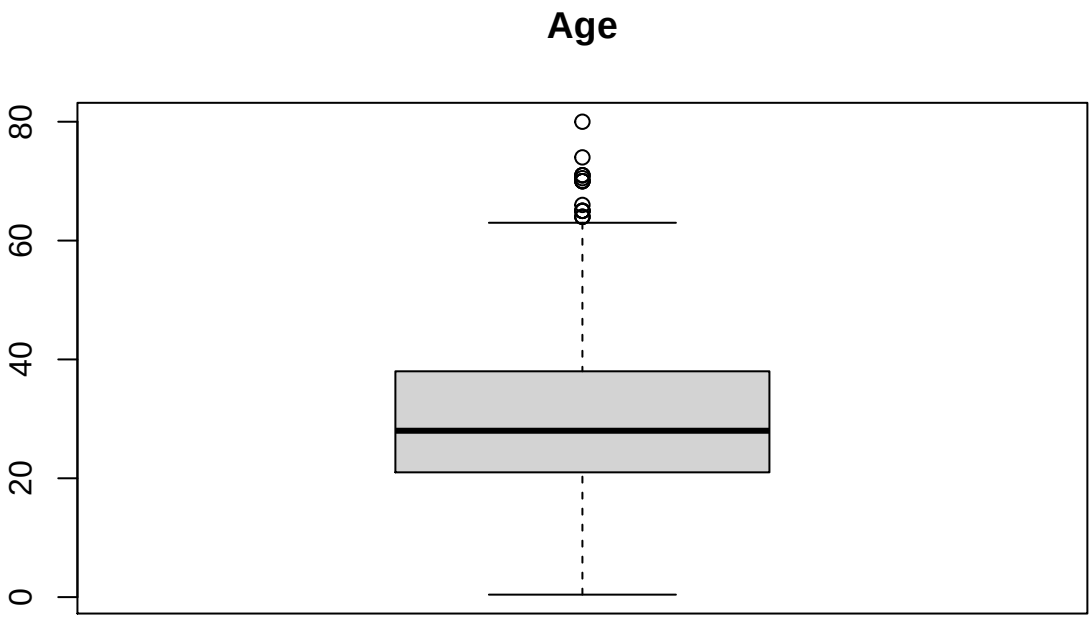
Ahora procedemos a normalizar los atributos. Como se pudo observar en el apartado anterior, solo el atributo fare es de tipo numeric, que se normalizará con la finalidad de eliminar o disminuir grandes cambios debido a la diferencia entre valores del mismo atributo. Para esto realizamos la siguiente operacion:

```
passengers$Fare <- scale(passengers$Fare)
```

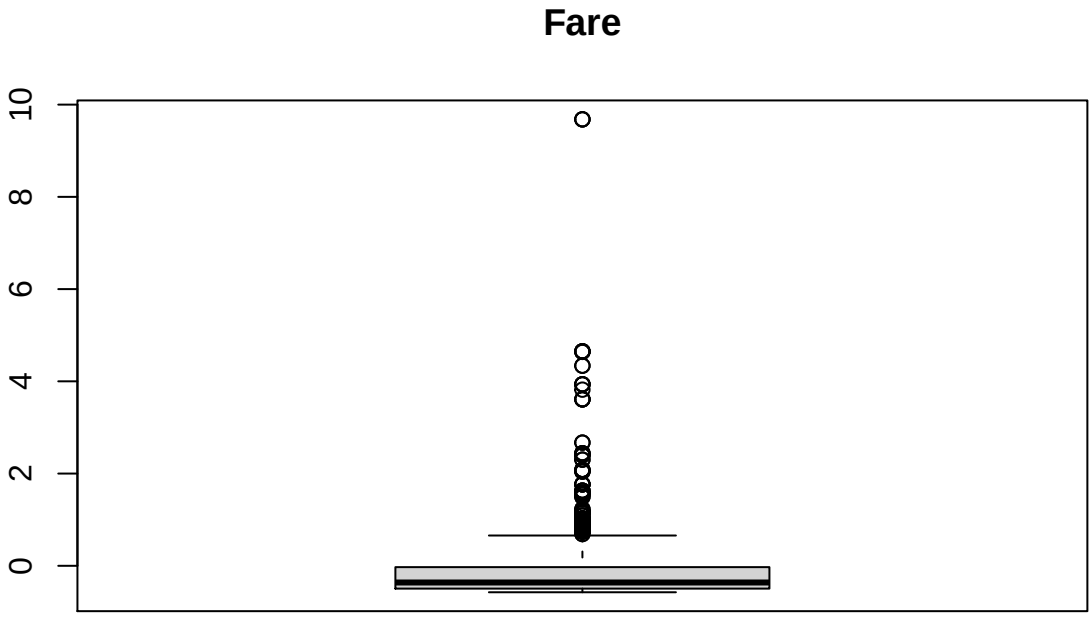
3.3 Outliers

En esta etapa trataremos de identificar outliers en nuestro data set. A continuacion analizamos los atributos age y fare ya que los demos atributos son de tipo categorico. Para esto creamos un boxplot para los atributos mencionados.

```
AgeOutliers<-boxplot(passengers$Age,main="Age")
```

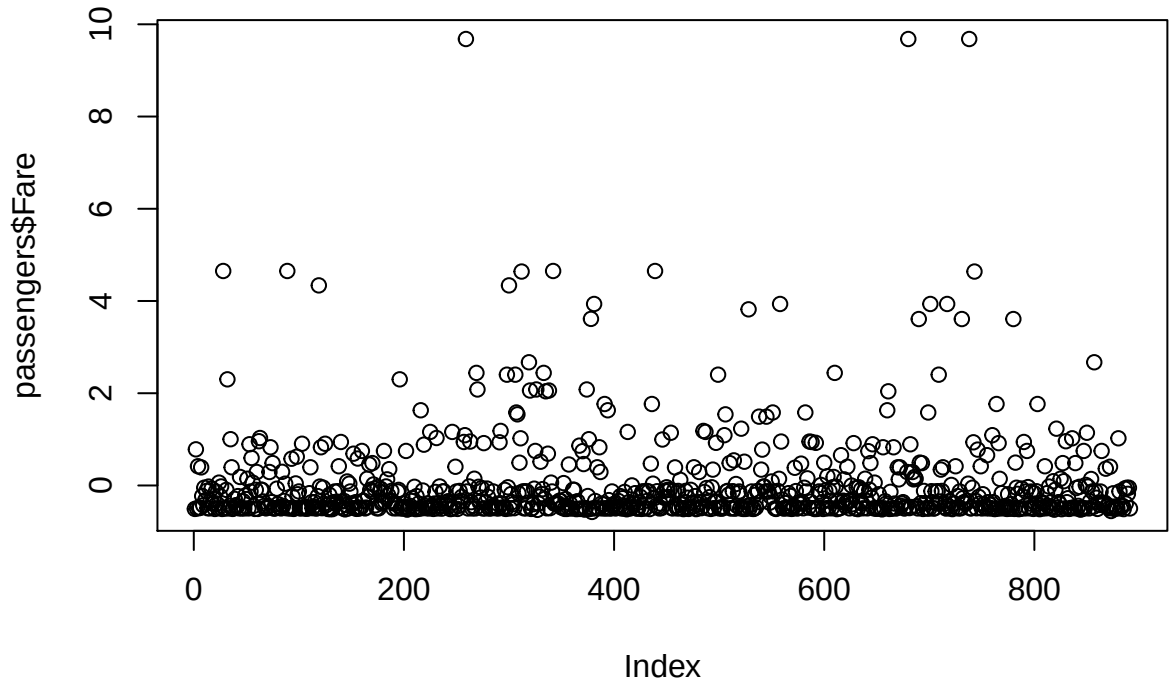



```
FareOutliers<-boxplot(passengers$Fare,main="Fare")
```



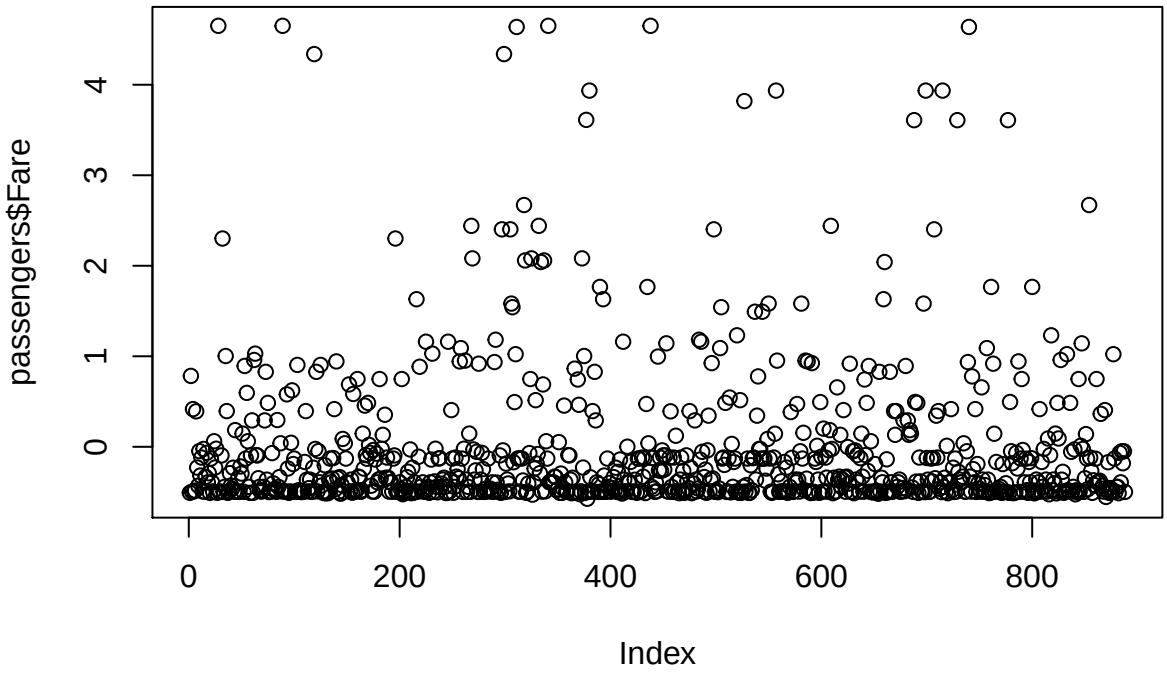
En el caso de el atributo Age podemos observar que, se considera como outlier a las personas mayores a 65 años aproximadamente. Esto se debe a que, como se menciono anteriormente, la mayoria de pasajerons tenia una edad entre 20-30 años. En este caso no se realizara ninguna operacion ya que esta informacion es real y muy importante para el analisis. Por otro lado, con el atributo Fare si podemos distinguir claramente 3 grupos de outliers que a priori no se pueden considerar normales. Para visualizar los registros de mejor manera ploteamos este atributo de la siguiente manera:

```
plot(passengers$Fare)
```



A pesar de que no se tiene evidencia para asumir que los valores son reales, alterados o erroneos, procederemos a eliminarlos. Esto se lleva acabo con la finalidad de eliminar 3 registros que alteran la distribucion de los datos y por su cantidad su eliminacion no representa una perdida significante de informacion. Para eliminar los outliers procedemos a aplicar lo siguiente:

```r
#Reemplazamos nuestro dataset, donde ahora se filtraran los valores que no cumplan la condicion (los 3 r
passengers <- filter(passengers, Fare<8)
#Ploteamos el atributo Fare del data set temporal
plot(passengers$Fare)
```



3.4 Analisis de componentes

Ahora que tenemos un data set "limpio", procederemos a analizar los atributos para determinar los componentes principales con la funcion *prcomp*.

```r
passengers.pca <- prcomp(passengers[,c(1,2,4,5,6,7)],na.rn=TRUE, scale = TRUE)
```

```
## Warning: In prcomp.default(passengers[, c(1, 2, 4, 5, 6, 7)], na.rn = TRUE,
##     scale = TRUE) :
##  extra argument 'na.rn' will be disregarded
```

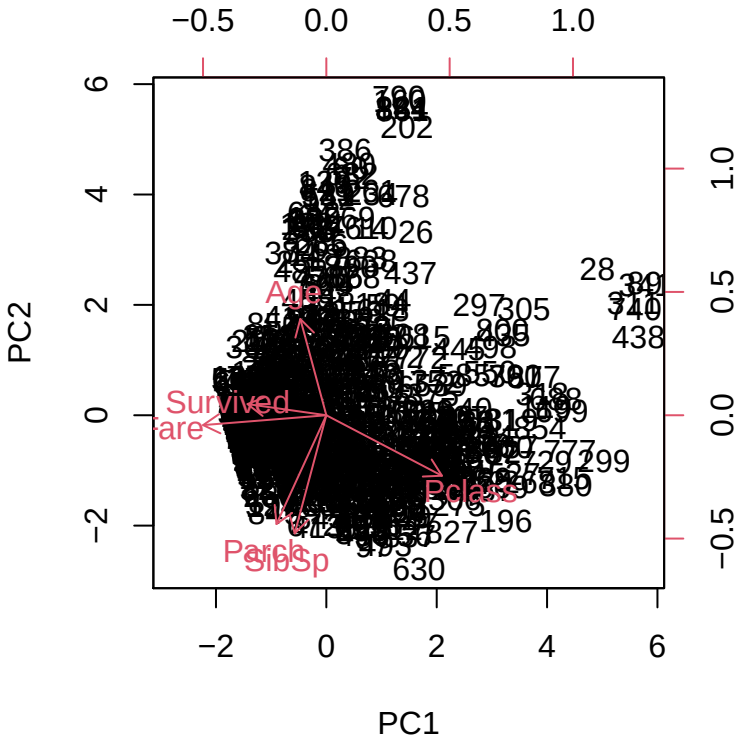
```r
passengers.pca $rotation <- -1*passengers.pca $rotation
passengers.pca
```

```
## Standard deviations (1, .., p=6):
## [1] 1.3833458 1.2730747 0.9837499 0.8054998 0.7427308 0.5453368
##
## Rotation (n x k) = (6 x 6):
##                PC1         PC2         PC3        PC4         PC5        PC6
## Survived -0.3994202  0.05652932 -0.71529564  0.4382507  0.34369807  0.1242096
## Pclass    0.5860537 -0.30656422 -0.05145768  0.2406666  0.06246848  0.7057541
## Age      -0.1318453  0.48760386  0.59950362  0.5308577  0.27981854  0.1592054
## SibSp    -0.1603597 -0.59882902  0.27659492 -0.1400612  0.71042569 -0.1219098
## Parch    -0.2526336 -0.55133565  0.17218716  0.5652974 -0.50568605 -0.1651586
## Fare     -0.6245658 -0.04998088  0.14193834 -0.3592043 -0.19811076  0.6473001
```

Como podemos observar, en el componenete PC1 y PC6 se muestra que practicamente toda la variacion se encuentra dada por el atributo PClass. En el caso del componente PC2, PC3 y PC4 la edad provee la variacion. Para el componente PC5 la variacion se encuetra determinada por el atributo SibSp. Finalmente, podemos ver que en el PC6 ademas de la edad el atributo Fare hace varia el atributo.

Para visualizar esto resultados aplicamos el siguiente comando:

```
biplot(passengers.pca, scale = 0)
```



Como podemos observar, el atributo Pclass toma los valores mas alejados en el eje de la PC1 y por otro lado la edad en el eje Y de la PC2. De esta manera se representa el emfasis que tiene cada atributo en la PCs.

4. Análisis de los datos.

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

De acuerdo la información procesada anteriormente y en especial al PCA vamos a trabajar con los atributos: Survived, Pclass, Sex, Age y Fare.

No vamos a trabajar con SibSp porque los datos son confusos al no aclarar específicamente si corresponden a hermanos o conyuges e igual para Parch. También para nosotros no tiene sentido evaluar el lugar de embarque porque esto no va a repercutir en el resultado final de supervivencia.

```
passengers$SibSp <- NULL
passengers$Parch <- NULL
passengers$Embarked <- NULL
```

Adicionalmente, vamos a agrupar algunas variables para facilitar su comprensión al momento de ralizar pruebas de hipótesis.

Separamos el genero en hombres y mujeres:

```
passengers$hombres <- passengers$Sex == "male"
passengers$mujeres <- passengers$Sex == "female"
```

También podemos segmentar la clase del tiquete para que a futuro nos ayude a comprender que clase social tuvo más sobrevivientes:

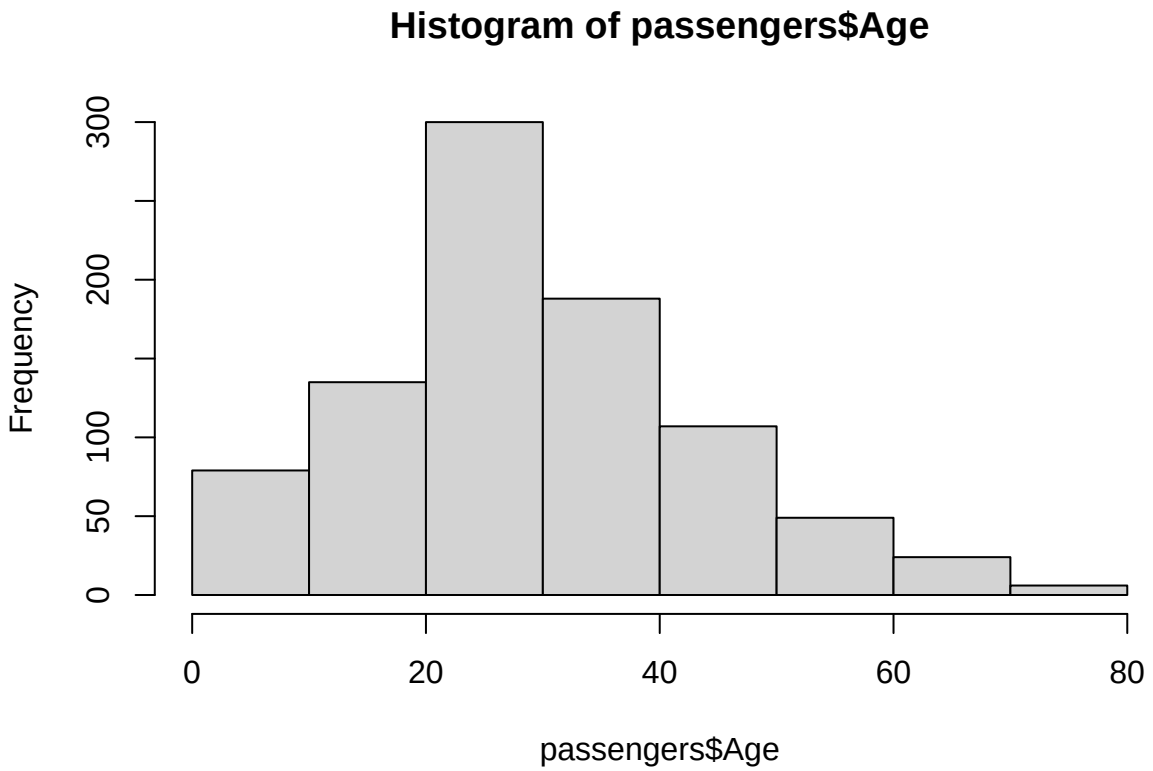
```
passengers$Clase_Alta <- passengers$Pclass == 1
passengers$Clase_Media <- passengers$Pclass == 2
passengers$Clase_Baja <- passengers$Pclass == 3
```

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

De acuerdo con las variables con las que hemos decidido trabajar, la única a la cual se le puede evaluar la normalidad es a la de la edad "Age". Por esto mismo, no tiene sentido evaluar la homogeneidad de la varianza para solo un atributo.

Ahora vamos a comprobar la normalidad de la variabale cuantitativa Age. Primero vamos a ver con un histograma como se ve la distribución de los datos:

```
# Histograma de la Edad
hist(passengers$Age)
```



Según el gráfico, aparentemente se ve un distribución uniforme pero no normal. Debido a esto y para estar totalmente seguros, vamos a aplicar 3 tests de normalidad. Para la tres pruebas se realiza el siguiente planteamiento:

H0: El atributo de la Edad distribuye normal H1: Se rechaza la hipótesis nula si el pvalue es menor al nivel de significancia de 0.05

Shapiro-Wilk:

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

```
##
##  Shapiro-Wilk normality test
##
## data:  passengers$Age
## W = 0.97908, p-value = 5.618e-10
```

El valor de probabilidad es menor a 0.05 por lo que podemos decir que nuestros datos NO siguen una distribución normal.

Lilliefors (Kolmogorov-Smirnov)

```
lillie.test(passengers$Age)
```

```
##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  passengers$Age
## D = 0.072614, p-value = 5.744e-12
```

El valor de p también es menor a 0.05, por lo que podemos decir que nuestros datos NO siguen una distribución normal.

Anderson-Darling

```
ad.test(passengers$Age)
```

```
##
##  Anderson-Darling normality test
##
## data:  passengers$Age
## A = 5.5978, p-value = 8.407e-14
```

Para esta última prueba también el valor de p es menor a 0.05.

Después de relizar los 3 tests de normalidad, existe suficiente evidencia estadística para confirmar que los datos de la variable Age no tienen una distribución normal sino uniforme.

4.3. Aplicación de pruebas estadísticas para comparar los grupos de datos. En función de los datos y el objetivo del estudio, aplicar pruebas de contraste de hipótesis, correlaciones, regresiones, etc. Aplicar al menos tres métodos de análisis diferentes

Primero creamos nuestros datasets de entrenamiento y test con una relación de 80% y 20% respectivamente

```
train<-passengers[1:710,]
test<-passengers[711:888,]
```

Modelo de Regresión Logística Multiple con variables explicativas cualitativas y cuantitativas:

Se realizará una Regresión Logística, donde usaremos los datasets de entrenamiento y prueba creados anteriormente y como variable de clasificación el atributo "Survived":

```
logmodel_1 <- glm(formula=Survived~Pclass+Sex+Fare+Age, data=train, family=binomial)
summary(logmodel_1)
```

```
##
## Call:
## glm(formula = Survived ~ Pclass + Sex + Fare + Age, family = binomial,
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3430  -0.6888  -0.4335   0.6412   2.4026
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.377003   0.525084   8.336  < 2e-16 ***
## Pclass      -1.165386   0.158854  -7.336  2.2e-13 ***
## Sexmale     -2.637172   0.209067 -12.614  < 2e-16 ***
## Fare        -0.204656   0.147238  -1.390  0.16454
## Age         -0.020708   0.007133  -2.903  0.00369 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 948.81  on 709  degrees of freedom
## Residual deviance: 659.25  on 705  degrees of freedom
## AIC: 669.25
##
## Number of Fisher Scoring iterations: 4
```

Se realizan las predicciones con las cuales posteriormente se evaluará la eficacia del modelo:

```
predicciones = predict(logmodel_1, test)
```

Modelo de Arbol de Decisión:

Para este modelo vamos a usar los mismos atributos aplicados al modelo regresión logística. Se emplearán los mismos datasets de entrenamiento y prueba aplicados al modelo anterior

```
# Creación del modelo
model_2 <-rpart(formula=Survived~Pclass+Sex+Fare+Age, data=train)
summary(model_2)
```

```
## Call:
## rpart(formula = Survived ~ Pclass + Sex + Fare + Age, data = train)
```

```
##   n= 710
##
##           CP nsplit rel error    xerror       xstd
## 1 0.29297266      0 1.0000000 1.0045510 0.01724595
## 2 0.07143574      1 0.7070273 0.7120066 0.03699326
## 3 0.02100634      2 0.6355916 0.6428270 0.03524460
## 4 0.02018307      3 0.6145853 0.6507207 0.03569390
## 5 0.01355427      4 0.5944022 0.6322091 0.03589869
## 6 0.01169843      6 0.5672937 0.6462606 0.03822154
## 7 0.01074461      8 0.5438968 0.6384360 0.03838331
## 8 0.01000000     10 0.5224076 0.6325254 0.03826383
##
## Variable importance
##    Sex   Fare Pclass    Age
##     51     21     19     10
##
## Node number 1: 710 observations,    complexity param=0.2929727
##   mean=0.3887324, MSE=0.2376195
##   left son=2 (455 obs) right son=3 (255 obs)
##   Primary splits:
##       Sex    splits as  RL, improve=0.29297270, (0 missing)
##       Pclass < 2.5        to the right, improve=0.09331851, (0 missing)
##       Fare   < -0.3485058 to the left,  improve=0.07552790, (0 missing)
##       Age    < 5.5        to the right, improve=0.01271085, (0 missing)
##   Surrogate splits:
##       Fare < 0.9113299  to the left,  agree=0.668, adj=0.075, (0 split)
##       Age  < 15.5       to the right, agree=0.649, adj=0.024, (0 split)
##
## Node number 2: 455 observations,    complexity param=0.02100634
##   mean=0.1912088, MSE=0.154648
##   left son=4 (354 obs) right son=5 (101 obs)
##   Primary splits:
##       Pclass < 1.5        to the right, improve=0.05036573, (0 missing)
##       Fare   < -0.1246635 to the left,  improve=0.04950912, (0 missing)
##       Age    < 13.5       to the right, improve=0.02745961, (0 missing)
##   Surrogate splits:
##       Fare < -0.1246635 to the left,  agree=0.899, adj=0.545, (0 split)
##       Age  < 55.75      to the left,  agree=0.787, adj=0.040, (0 split)
##
## Node number 3: 255 observations,    complexity param=0.07143574
##   mean=0.7411765, MSE=0.1918339
##   left son=6 (121 obs) right son=7 (134 obs)
##   Primary splits:
##       Pclass < 2.5        to the right, improve=0.24637150, (0 missing)
##       Fare   < 0.3177668  to the left,  improve=0.08967708, (0 missing)
##       Age    < 47.5       to the left,  improve=0.02396543, (0 missing)
##   Surrogate splits:
##       Fare < -0.1354644 to the left,  agree=0.808, adj=0.595, (0 split)
##       Age  < 22.5       to the left,  agree=0.608, adj=0.174, (0 split)
##
## Node number 4: 354 observations,    complexity param=0.01355427
##   mean=0.1440678, MSE=0.1233123
##   left son=8 (323 obs) right son=9 (31 obs)
##   Primary splits:
```

```
##       Age    < 13.5       to the right, improve=0.045969420, (0 missing)
##       Fare   < -0.4950159 to the left,  improve=0.017586370, (0 missing)
##       Pclass < 2.5        to the right, improve=0.001091438, (0 missing)
##
## Node number 5: 101 observations
##   mean=0.3564356, MSE=0.2293893
##
## Node number 6: 121 observations,    complexity param=0.02018307
##   mean=0.5123967, MSE=0.2498463
##   left son=12 (18 obs) right son=13 (103 obs)
##   Primary splits:
##       Fare < -0.1541249 to the right, improve=0.1126340, (0 missing)
##       Age  < 46         to the left,  improve=0.0372565, (0 missing)
##
## Node number 7: 134 observations
##   mean=0.9477612, MSE=0.04950991
##
## Node number 8: 323 observations
##   mean=0.120743, MSE=0.1061642
##
## Node number 9: 31 observations,    complexity param=0.01355427
##   mean=0.3870968, MSE=0.2372529
##   left son=18 (23 obs) right son=19 (8 obs)
##   Primary splits:
##       Pclass < 2.5        to the right, improve=0.34899410, (0 missing)
##       Age    < 7.5        to the left,  improve=0.08297578, (0 missing)
##       Fare   < -0.4013765 to the left,  improve=0.07333438, (0 missing)
##
## Node number 12: 18 observations
##   mean=0.1111111, MSE=0.09876543
##
## Node number 13: 103 observations,    complexity param=0.01169843
##   mean=0.5825243, MSE=0.2431897
##   left son=26 (62 obs) right son=27 (41 obs)
##   Primary splits:
##       Fare < -0.4924095 to the right, improve=0.06051818, (0 missing)
##       Age  < 6.5        to the right, improve=0.04128039, (0 missing)
##   Surrogate splits:
##       Age < 41.5       to the left,  agree=0.66, adj=0.146, (0 split)
##
## Node number 18: 23 observations
##   mean=0.2173913, MSE=0.1701323
##
## Node number 19: 8 observations
##   mean=0.875, MSE=0.109375
##
## Node number 26: 62 observations,    complexity param=0.01169843
##   mean=0.483871, MSE=0.2497399
##   left son=52 (15 obs) right son=53 (47 obs)
##   Primary splits:
##       Fare < -0.4367649 to the left,  improve=0.1570272, (0 missing)
##       Age  < 6.5        to the right, improve=0.1173611, (0 missing)
##
## Node number 27: 41 observations
```

```
##   mean=0.7317073, MSE=0.1963117
##
## Node number 52: 15 observations
##   mean=0.1333333, MSE=0.1155556
##
## Node number 53: 47 observations,    complexity param=0.01074461
##   mean=0.5957447, MSE=0.240833
##   left son=106 (36 obs) right son=107 (11 obs)
##   Primary splits:
##       Age  < 6.5        to the right, improve=0.12457280, (0 missing)
##       Fare < -0.3444711 to the left,  improve=0.05366617, (0 missing)
##
## Node number 106: 36 observations,    complexity param=0.01074461
##   mean=0.5, MSE=0.25
##   left son=212 (10 obs) right son=213 (26 obs)
##   Primary splits:
##       Age  < 22.5       to the left,  improve=0.24615380, (0 missing)
##       Fare < -0.3545578 to the left,  improve=0.09090909, (0 missing)
##
## Node number 107: 11 observations
##   mean=0.9090909, MSE=0.08264463
##
## Node number 212: 10 observations
##   mean=0.1, MSE=0.09
##
## Node number 213: 26 observations
##   mean=0.6538462, MSE=0.2263314
```

Ahora realizamos las predicciones para evaluar el árbol a futuro:

```
predicciones2 <- predict(model_2, test)
```

Modelo de Random Forest

Con este modelo se generan múltiples árboles. Cada árbol entrega classificación (vota por un atributo). Y el resultado es la atributo con mayor número de votos en todo el bosque (forest)

```
newdat <- Survived ~ Pclass + Sex + Age+Fare
model_3 <- randomForest(newdat, data = train, importance=TRUE, ntree=2000)
```

Ahora conoceremos las variables mas importantes segun este modelo

```
importance(model_3)
```

```
##          %IncMSE IncNodePurity
## Pclass  69.50700     11.175064
## Sex    122.90674     36.353317
## Age     28.18694      9.367079
## Fare    38.57358     14.234748
```

Ahora realizamos las predicciones para evaluar el modelo Randon Forest a futuro:

```
predicciones3 <- predict(model_3, test)
```

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

Cálculo de los OR para el modelo de regresión logística multiple:

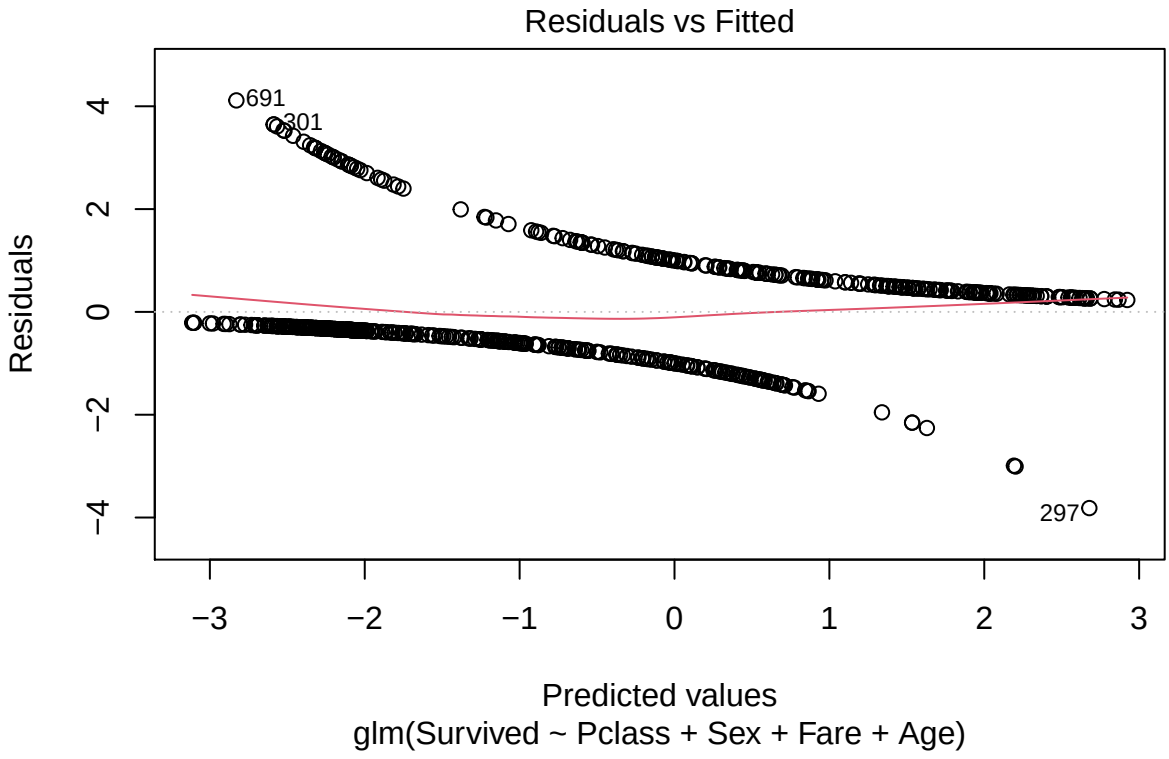
```
exp(cbind(coef(logmodel_1),confint(logmodel_1)))
```

```
## Waiting for profiling to be done...
```

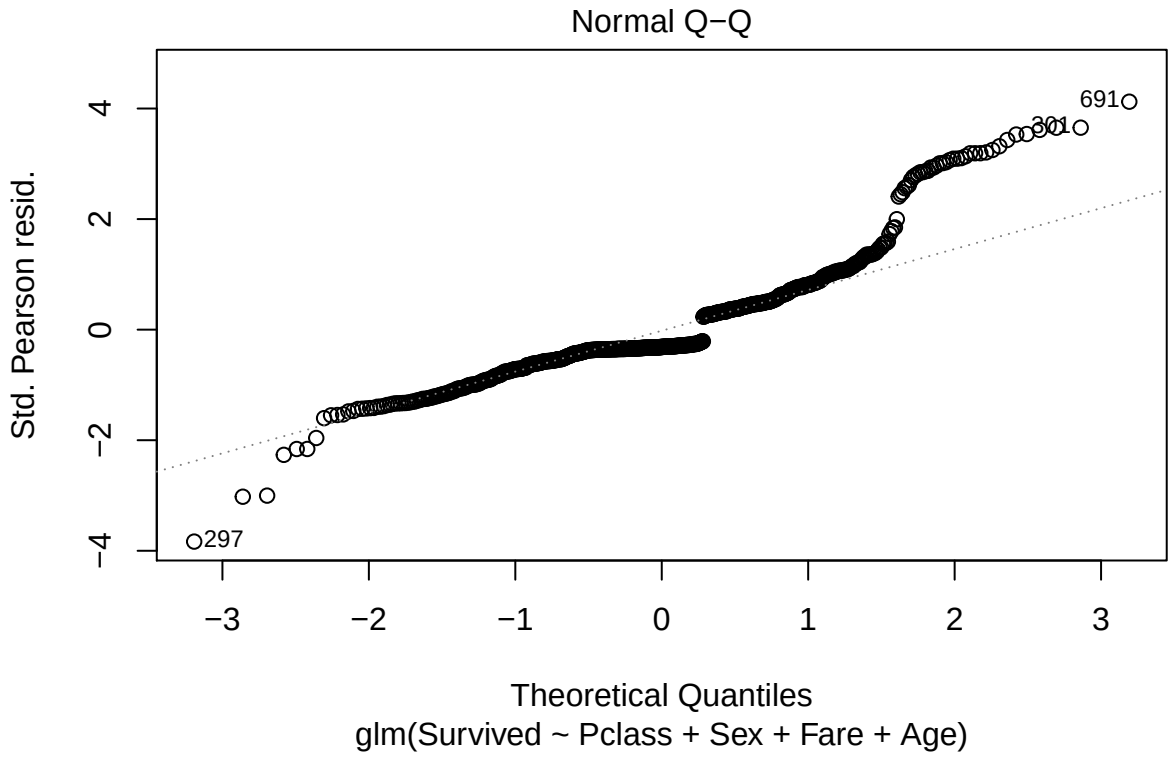
```
##                               2.5 %      97.5 %
## (Intercept) 79.59915303 29.10579309 228.6135872
## Pclass       0.31180243  0.22708015   0.4236960
## Sexmale      0.07156334  0.04703531   0.1068710
## Fare         0.81492792  0.61042505   1.0886349
## Age          0.97950472  0.96572418   0.9931509
```

De acuerdo a los OR obtenidos, se tiene evidencia estadística para afirmar que para el modelo logístico multiple no existe un unico atributo que afecte de manera crítica a este modelo.

```
plot(logmodel_1,1)
```




```
plot(logmodel_1,2)
```

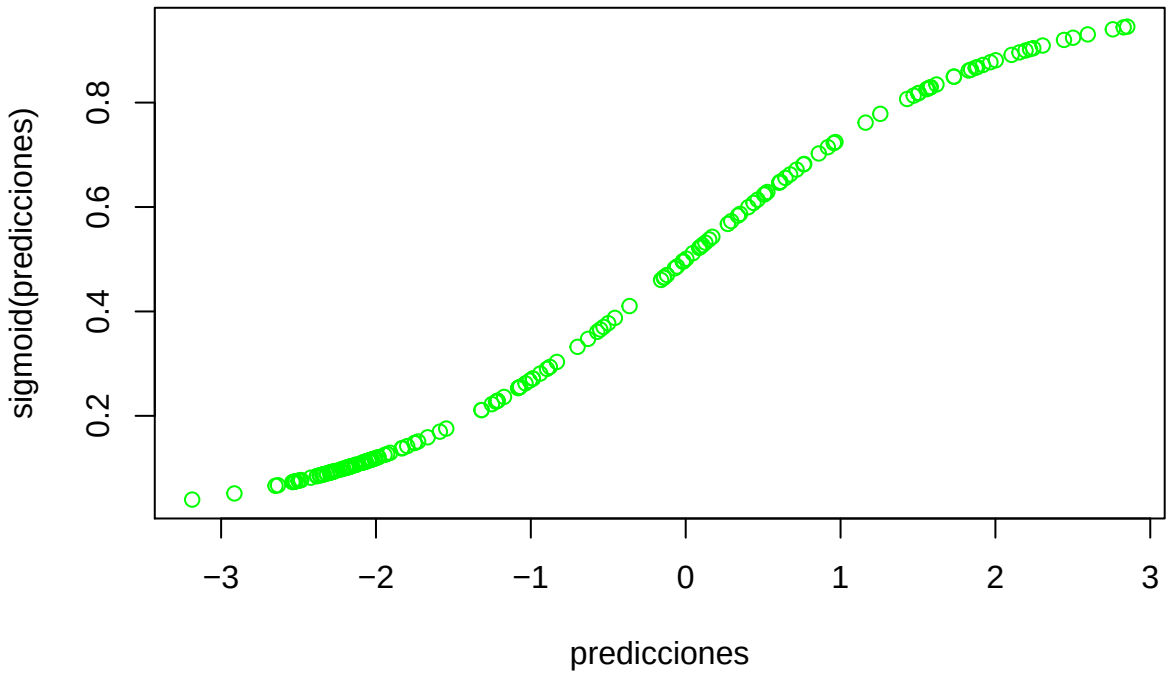


Se puede inferir que las varianzas del modelo NO son constantes ya que los puntos se estan distribuyendo de manera aleatoria. A partir del gráfico cuantil-cuantil, se puede afirmar que los residuales del modelo no distribuyen normal

Gráfico del modelo de regresión logistíca multiple:

Primero creamos el plano donde vamos a poner los puntos de la predicción hecha, para ello usamos la función sigmoid:

```
sigmoid = function(x){1/(1+exp(-x))}
x<-seq(-5,5,0.02)
plot(predicciones, sigmoid(predicciones),col="Green")
```



Revisamos las predicciones hechas:

```
view(predicciones)
```

Ahora vamos a comprobar la precisión del modelo, para esto primero aproximamos los valores que se predijeron a 1 ó 0, es decir, a sobrevivió o no.

```
newpred <- ifelse(predicciones>0.5,1,0)
```

Ahora creamos una matriz de confusión para evaluar el rendimiento del modelo de entrenamiento versus el del test.Antes se debe convertir a factor el atributo survived del test y las predicciones aproximadas:

```
#Conversiones a factor
test$Survived <- factor(test$Survived)
newpred <- factor(newpred)
# Matriz de Confusión
confusionMatrix(newpred, test$Survived)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 108  23
##          1   7  40
##
##                Accuracy : 0.8315
##                  95% CI : (0.7682, 0.8833)
##     No Information Rate : 0.6461
##     P-Value [Acc > NIR] : 3.515e-08
##
##                   Kappa : 0.609
```

```
##
##  Mcnemar's Test P-Value : 0.00617
##
##             Sensitivity : 0.9391
##             Specificity : 0.6349
##          Pos Pred Value : 0.8244
##          Neg Pred Value : 0.8511
##              Prevalence : 0.6461
##          Detection Rate : 0.6067
##    Detection Prevalence : 0.7360
##       Balanced Accuracy : 0.7870
##
##        'Positive' Class : 0
##
```

La precisión del modelo es 83.15%

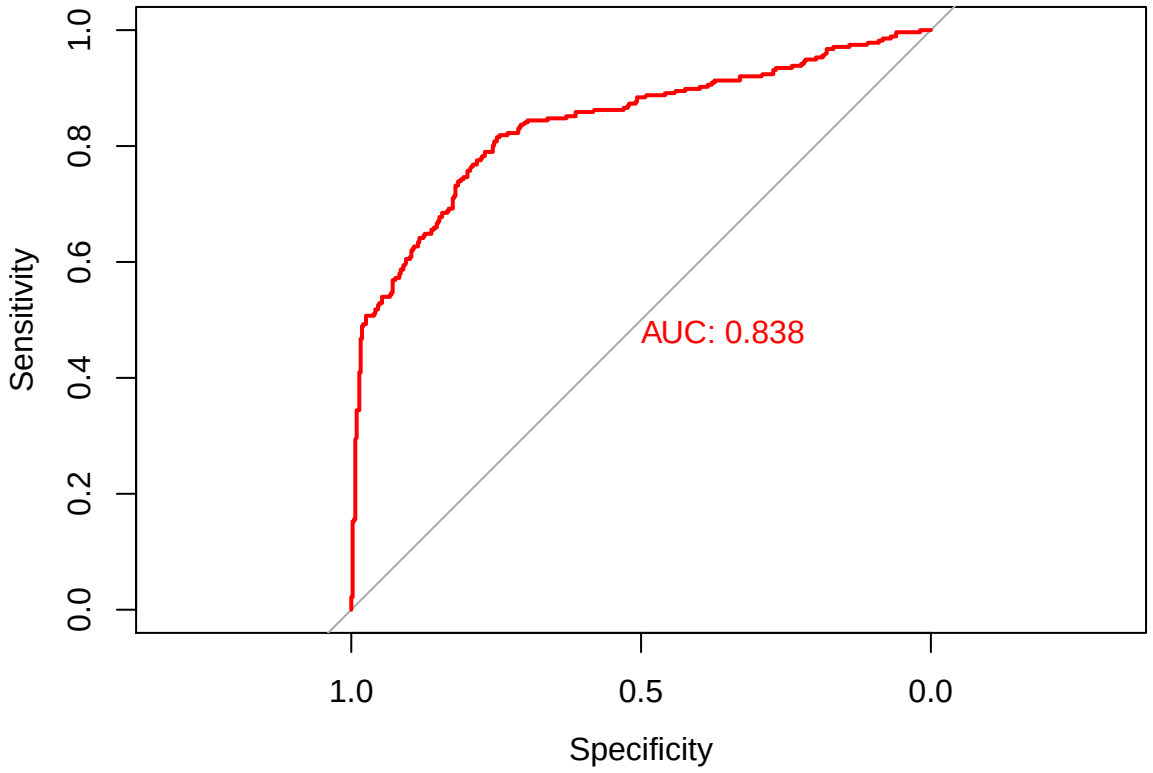
Ahora también podemos ver el comportamiento del modelo en un Curva ROC, otra medida para evaluar un modelo logístico:

```
## Curva ROC:
prob=predict(logmodel_1, train, type="response")
r=roc(train$Survived,prob, data=train)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(r, col="red", print.auc=TRUE)
```



Entre más se acerque la curva a 1 mejor es el modelo. Para este caso aunque no es perfecto el modelo comprende la mayor parte del área bajo la curva

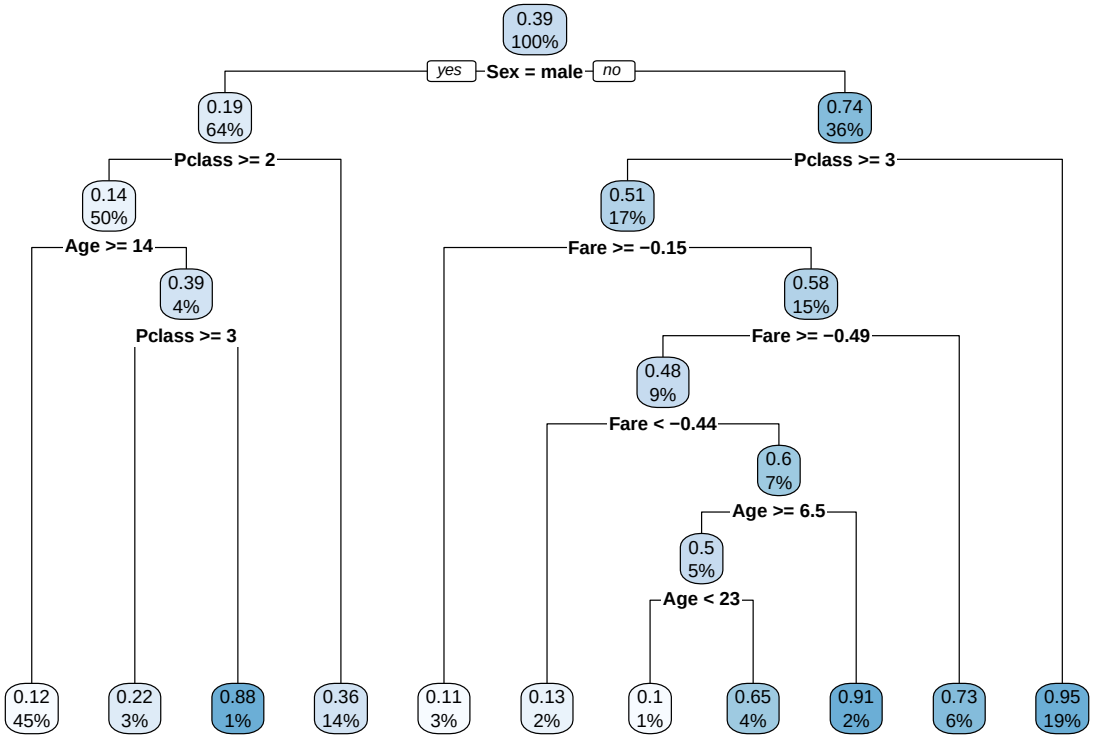
Gráfico del árbol de decisión

Vamos a realizar el gráfico y a comprobar la precisión del modelo, para esto primero aproximamos los valores que se predijeron a 1 ó 0, es decir, a sobrevivió o no.

```
# Aproximación de los valores predecidos
newpred_model2 <- ifelse(predicciones2>0.5,1,0)
# vista de las nueva predicción
View(newpred_model2)
```

Ahora graficamos el modelo:

```
rpart.plot(model_2)
```



Ahora creamos una matriz de confusión para evaluar el rendimiento del modelo de entrenamiento versus el del test.Antes se debe convertir a factor las predicciones aproximadas:

```
#Conversiones a factor
newpred_model2 <- factor(newpred_model2)
test$Survived <- factor(test$Survived)
# Matriz de Confusión
confusionMatrix(newpred_model2, test$Survived)
```

```
## Confusion Matrix and Statistics
##
```

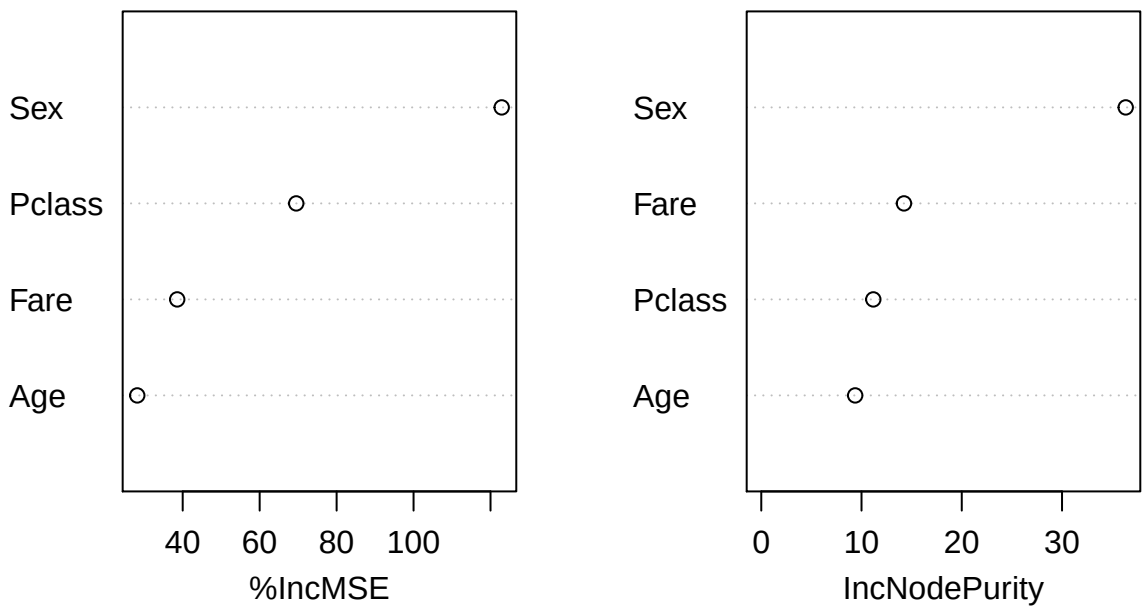
```
##           Reference
## Prediction   0   1
##          0 107  20
##          1   8  43
##
##                Accuracy : 0.8427
##                  95% CI : (0.7807, 0.8929)
##     No Information Rate : 0.6461
##     P-Value [Acc > NIR] : 4.368e-09
##
##                   Kappa : 0.6406
##
##  Mcnemar's Test P-Value : 0.03764
##
##             Sensitivity : 0.9304
##             Specificity : 0.6825
##          Pos Pred Value : 0.8425
##          Neg Pred Value : 0.8431
##              Prevalence : 0.6461
##          Detection Rate : 0.6011
##    Detection Prevalence : 0.7135
##       Balanced Accuracy : 0.8065
##
##        'Positive' Class : 0
##
```

Como podemos observar el modelo de árbol de decisión tiene una precisión mas alta que le de regresió logística, esta es de 84.27%.

Random Forest

Vamos a visualizar la importancia de las variables según este modelo:

```
varImpPlot(model_3, main = "RF: Importancia de las Variables")
```



Se puede ver que para este modelo prima el Sexo, luego la clase, luego el costo del tiquete y después la edad

Ahora vamos a comprobar la precisión del modelo, para esto primero aproximamos los valores que se predijeron a 1 ó 0, es decir, a sobrevivió o no.

```
# Aproximación de los valores predecidos
newpred_model3 <- ifelse(predicciones3>0.5,1,0)
```

Ahora creamos una matriz de confusión para evaluar el rendimiento del modelo de entrenamiento versus el del test.Antes se debe convertir a factor el atributo survived del test y las predicciones aproximadas:

```
#Conversiones a factor
test$Survived <- factor(test$Survived)
newpred_model3 <- factor(newpred_model3)
# Matriz de Confusión
confusionMatrix(newpred_model3, test$Survived)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 110  22
##          1   5  41
##
##                Accuracy : 0.8483
##                  95% CI : (0.787, 0.8976)
##     No Information Rate : 0.6461
##     P-Value [Acc > NIR] : 1.449e-09
```

```
##
##                   Kappa : 0.6468
##
##  Mcnemar's Test P-Value : 0.002076
##
##             Sensitivity : 0.9565
##             Specificity : 0.6508
##          Pos Pred Value : 0.8333
##          Neg Pred Value : 0.8913
##              Prevalence : 0.6461
##          Detection Rate : 0.6180
##    Detection Prevalence : 0.7416
##       Balanced Accuracy : 0.8037
##
##        'Positive' Class : 0
##
```

De acuerdo a los resultados obtenidos, el modelo que mejor explica la supervivencia en el Titanic es Random Forest con 84,83%.

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

Para concluir primero mencionamos que el data set contiene diferentes atributos que no son útiles para crear un modelo, por lo que fueron eliminados.Además, el data set contiene diferentes anomalías que fueron corregidas durante el proceso de limpieza de datos entre ellos: valores nulos, outliers etc.

Una vez que el data set fue "limpiado" y analizado visualmente se implementaron los modelos supervisados que nos dieron los siguientes resultados: el modelo de regresión múltiple nos da una precisión de 83.15%, el árbol de decisión un 84.27% y el modelo de Random Forest un 84.83%. A pesar de que este último modelo presenta una mejor precisión, los otros dos presentan medidas muy similares. Esto sumado a la poca cantidad de registros tanto para el entrenamiento como para el testeo de los modelos hace muy complicado afirmar que el Rain Forest es el modelo más óptimo.

Por otro lado, como se puede observar en el gráfico del modelo de árbol de decisión, el atributo sexo es los atributos más decisivos al momento de realizar la predicción. Por ejemplo, del gráfico mencionado se puede extraer la siguiente regla de asociación: una MUJER con un ticket perteneciente a CLASE 3 tiene un 95% de sobrevivir al accidente. Otro atributo de gran impacto para la supervivencia de un pasajero, como era de esperarse, es la clase. A mayor clase mayor probabilidad de sobrevivir. Por ejemplo, un hombre mayor a 14 años y de clase 3 tenía un 88% de probabilidades de sobrevivir, pero si su clase era de 2 o menos, esta probabilidad era de apenas el 22%. De esta manera se demuestra que los atributos más relevantes son el sexo y la clase.

7. Tabla de contribuciones

CONTRIBUCIONES	FIRMA
Investigación Previa	G.C., I.O.
Redacción de las respuestas	G.C., I.O.
Desarrolo Código	G.C., I.O.

```r
write.csv(passengers, "titanic_clean.csv", row.names = FALSE)
```