Tipologia y Ciclo de vida de los Datos. Practica 2

Autores: Alexis Germán Arroyo Peña y Gabriel Pulido de Torres

Enero 2021

Contents

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

Hemos optado por analizar el dataset de datos del titanic.

En este conjunto de datos tenemos información acerca de los pasajeros que iban en el Titanic, naufragado en 1912 y en el que murieron 1500 personas. Además de información descriptiva del tipo de pasajero tenemos también el indicador de si sobrevivió al hundimiento o no. El objetivo es decidir si las variables aportadas son suficientes para crear un modelo predictivo que prediga la supervivencia o no de un pasajero.

Es un buen ejemplo de conjunto de datos que permiten entrenar modelos predictivos supervisados. Contiene un número manejable de datos y de atributos de distintos y variados tipos. A su vez es un conjunto que no está limpio (contiene nulos por ejemplo) lo que nos obligará a poner en práctica técnicas de limpieza de los datos.

Estructura del dataset

Realizamos un primer contacto con el conjunto de datos, visualizando su estructura.

```
# Cargamos el fichero de datos
data <- read.csv('train.csv',stringsAsFactors = FALSE, na.strings = "")
dim.data.frame(data)
```

```
## [1] 891  12
```

```
# Verificamos la estructura del conjunto de datos
str(data)
```

```
## '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  NA "C85" NA "C123" ...
##  $ Embarked   : chr  "S" "C" "S" "S" ...
```

El conjunto de datos incluye 12 variables y 891 observaciones.

Descripción del conjunto de datos:

- Passengerld: Contador de pasajeros del 1 al 891.
- Survived: esta variable toma dos valores e indica si el pasajero sobrevivió. (0= No, 1= Si).
- pClass: clase del ticket. 1=1st (clase alta), 2=2nd (clase media) y 3=3rd (clase baja).
- Name: nombre completo del pasajero.
- Sex: sexo del pasajero (Female o Male).
- Age: edad del pasajero
- SibSp: número de hermanos/hermanas, hermanastros/hermanastros y marido o esposa del pasajero que también iban a bordo.
- Parch: número de hijas, hijos, padre y madre del pasajero a bordo del Titanic.

- Ticket: El número del ticket del pasajero.
- Fare: Es la tarifa del pasajero en dólares.
- Cabin: Código identificativo de la cabina.
- Embarked: el puerto en el que embarcó el pasajero (C = Cherbourg, Q =Queenstown, S = Southampton).

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

El conjunto de datos que se proporciona en Kaggle está dividido en dos partes, un conjunto de datos de train, y un conjunto de datos de test. La diferencia principal es que mientras que en el conjunto de datos de train disponemos del valor de la variable Survived, en el de test no disponemos de esa variable.

Como parte de los modelos que vamos a diseñar son modelos supervisados, a la hora de validarlos no podemos usar el conjunto de datos de test, por lo que optamos trabajar sólamente con los de train sin fusionarlos.

En el apartado anterior leímos los datos en un dataframe (al que llamamos "data"). Recordamos su dimensionalidad y comprobamos los nombres y tipos de las variables.

```
## [1] 891  12
```

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

Aprovechamos para comprobar si hay registros duplicados usando el comando unique()

```
data_unique <- unique(data)
dim.data.frame(data_unique)
```

```
## [1] 891  12
```

```
remove(data_unique)
```

Se comprueba que la dimensionalidad ha quedado exactamente igual que cuando cargamos el dataset, por lo tanto, no hay registros duplicados (nos referimos a completos duplicados).

Mostramos una muestra de los primeros registros de nuestro dataframe:

```
summary(data)
```

```
##   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
##
```

```
df_status(data)
```

```
##        variable q_zeros p_zeros q_na  p_na q_inf p_inf      type unique
## 1   PassengerId       0    0.00    0  0.00     0     0   integer    891
## 2      Survived     549   61.62    0  0.00     0     0   integer      2
## 3        Pclass       0    0.00    0  0.00     0     0   integer      3
## 4          Name       0    0.00    0  0.00     0     0 character    891
## 5           Sex       0    0.00    0  0.00     0     0 character      2
## 6           Age       0    0.00  177 19.87     0     0   numeric     88
## 7         SibSp     608   68.24    0  0.00     0     0   integer      7
## 8         Parch     678   76.09    0  0.00     0     0   integer      7
## 9        Ticket       0    0.00    0  0.00     0     0 character    681
## 10         Fare      15    1.68    0  0.00     0     0   numeric    248
## 11        Cabin       0    0.00  687 77.10     0     0 character    147
## 12     Embarked       0    0.00    2  0.22     0     0 character      3
```

Hay algunas variables que carecen de interés por ser identificativas de cada registro. Se tratan de "PassengerId", "Ticket" y "Name". No nos interesa tener unívocamente identificado cada caso para realizar ningún tipo de análisis, no aportan nada.

2.1 PersonasTicket

Antes de eliminar la variable **Ticket**, nos va a servir para conocer el precio unitario que han pagado los pasajeros, ya que la variable **Fare** es la tarifa pagada en el ticket, pero dentro del mismo ticket pueden estar incluidas varias personas. Inicialmente vamos a obtener el número de personas que están incluidas en un mismo ticket.

```
nrow(table(data$Ticket, data$Fare))
```

```
## [1] 681
```

```
length(unique(data$Ticket))
```

```
## [1] 681
```

```
data %>% group_by(Ticket, Fare) %>% filter(row_number() == 1)
```

PassengerId	Survived	Pclass	Name
1	0	3	Braund, Mr. Owen Harris
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)
3	1	3	Heikkinen, Miss. Laina
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)
5	0	3	Allen, Mr. William Henry
6	0	3	Moran, Mr. James
7	0	1	McCarthy, Mr. Timothy J
8	0	3	Palsson, Master. Gosta Leonard
9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)
10	1	2	Nasser, Mrs. Nicholas (Adele Achem)
11	1	3	Sandstrom, Miss. Marguerite Rut
12	1	1	Bonnell, Miss. Elizabeth
13	0	3	Saundercock, Mr. William Henry
14	0	3	Andersson, Mr. Anders Johan
15	0	3	Vestrom, Miss. Hulda Amanda Adolfina
16	1	2	Hewlett, Mrs. (Mary D Kingcome)
17	0	3	Rice, Master. Eugene
18	1	2	Williams, Mr. Charles Eugene
19	0	3	Vander Planke, Mrs. Julius (Emelia Maria Vandemoortele)
20	1	3	Masselmani, Mrs. Fatima
21	0	2	Fynney, Mr. Joseph J
22	1	2	Beesley, Mr. Lawrence
23	1	3	McGowan, Miss. Anna "Annie"
24	1	1	Sloper, Mr. William Thompson
26	1	3	Asplund, Mrs. Carl Oscar (Selma Augusta Emilia Johansson)
27	0	3	Emir, Mr. Farred Chehab
28	0	1	Fortune, Mr. Charles Alexander
29	1	3	O'Dwyer, Miss. Ellen "Nellie"
30	0	3	Todoroff, Mr. Lalio
31	0	1	Uruchurtu, Don. Manuel E
32	1	1	Spencer, Mrs. William Augustus (Marie Eugenie)
33	1	3	Glynn, Miss. Mary Agatha
34	0	2	Wheadon, Mr. Edward H
35	0	1	Meyer, Mr. Edgar Joseph
36	0	1	Holverson, Mr. Alexander Oskar
37	1	3	Mamee, Mr. Hanna
38	0	3	Cann, Mr. Ernest Charles
39	0	3	Vander Planke, Miss. Augusta Maria
40	1	3	Nicola-Yarred, Miss. Jamila
41	0	3	Ahlin, Mrs. Johan (Johanna Persdotter Larsson)
42	0	2	Turpin, Mrs. William John Robert (Dorothy Ann Wonnacott)
43	0	3	Kraeff, Mr. Theodor
44	1	2	Laroche, Miss. Simonne Marie Anne Andree
45	1	3	Devaney, Miss. Margaret Delia
46	0	3	Rogers, Mr. William John
47	0	3	Lennon, Mr. Denis
48	1	3	O'Driscoll, Miss. Bridget
49	0	3	Samaan, Mr. Youssef
50	0	3	Arnold-Franchi, Mrs. Josef (Josefine Franchi)
51	0	3	Panula, Master. Juha Niilo
52	0	3	Nosworthy, Mr. Richard Cater
53	1	1	Harper, Mrs. Henry Sleeper (Myna Haxtun)
54	1	2	Faunthorpe, Mrs. Lizzie (Elizabeth Anne Wilkinson)
55	0	1	Ostby, Mr. Engelhart Cornelius
56	1	1	Woolner, Mr. Hugh
57	1	2	Rugg, Miss. Emily
58	0	3	Novel, Mr. Mansouer

Podemos obtener el precio por persona, dividiendo Fare / integrantes del ticket:

```
ticket_personas <- as.data.frame(data %>%
                                   group_by(Ticket) %>%
                                   dplyr::summarize(PersonasTicket=n()))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

Creamos un nuevo dataframe (llamado "ticket_personas") con las variables **Ticket** y **PersonasTicket**

```
df_status(ticket_personas)
```

```
##         variable q_zeros p_zeros q_na p_na q_inf p_inf      type unique
## 1         Ticket       0       0    0    0     0     0 character    681
## 2 PersonasTicket       0       0    0    0     0     0   integer      7
```

Este nuevo dataframe lo combinamos con nuestro dataframe original (un join) a través del número de ticket (variable **Ticket**) para poder incorporar el número de personas al dataframe original. Después de juntarlos eliminamos el df ticket_personas pues no lo necesitaremos.

```
data <- merge(data, ticket_personas, by = "Ticket")
remove(ticket_personas)
```

Ahora tenemos en nuestro dataframe original "data" una nueva variable "PersonasTicket". Esa variable será utilizada para dividir la tarifa del ticket "Fare" entre esta nueva columna incorporada. Con ello obtenemos una nueva variable que le llamamos "Price" y es el precio por persona que se paga en el billete (obteniendo un precio por persona lineal por billete).

```
data$Price <- data$Fare / data$PersonasTicket
```

Ahora ya podemos eliminar todas las variables que no vamos a necesitar: PassengerId, Name, Ticket, PersonasTicket:

```
data <- select(data, -PassengerId, -Name, -Ticket, -PersonasTicket)
```

2.2 FamilySize

Revisando **Parch** (número de Padres / hijos a bordo) y **SibSp** (sibling: número de hermanos, hermanas, hermanastros y hermanastras del pasajero, spouse: Marido o mujer del pasajero en el titanic) vemos que las podemos agregar en una nueva variable **FamilySize** que será la suma de estas dos variables más el propio viajero en cuestión. Por ello esta variable como mínimo valdrá uno.

```
data$FamilySize = data$SibSp + data$Parch + 1
```

2.3 Revisión y conversion de tipos

Tenemos algunas variables que, aunque a priori aparecen como "numeric" o "character" deberíamos convertir a "factor". Estas variables son: **Survived**, **Pclass**, **Embarked** y **Sex**. Tienen un número finito de valores y aunque puedan ser numéricas, ese número no nos aporta información con lo que los discretizamos convirtiendolos en factores:

```
data$Sex <- as.factor(data$Sex)
data$Survived <- as.factor(data$Survived)
data$Pclass <- as.factor(data$Pclass)
data$Embarked <- as.factor(data$Embarked)
```

La variable **Cabin** es de tipo character, pero como vamos a eliminarla no haremos ninguna conversión.

Finalmente nos quedarían como variables numéricas: **Age**, **SubSp**, **Parch**, **FamilySize**, **Fare** y **Price**

```
summary(data)
```

```
##  Survived Pclass      Sex           Age            SibSp           Parch
##  0:549    1:216   female:314   Min.   : 0.42   Min.   :0.000   Min.   :0.0000
##  1:342    2:184   male  :577   1st Qu.:20.12   1st Qu.:0.000   1st Qu.:0.0000
##           3:491                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
##       Fare           Cabin           Embarked      Price
##  Min.   :  0.00   Length:891         C   :168   Min.   :  0.000
##  1st Qu.:  7.91   Class :character   Q   : 77   1st Qu.:  7.763
##  Median : 14.45   Mode  :character   S   :644   Median :  8.850
##  Mean   : 32.20                      NA's:  2   Mean   : 17.789
##  3rd Qu.: 31.00                                 3rd Qu.: 24.288
##  Max.   :512.33                                 Max.   :221.779
##
##    FamilySize
##  Min.   : 1.000
##  1st Qu.: 1.000
##  Median : 1.000
##  Mean   : 1.905
##  3rd Qu.: 2.000
##  Max.   :11.000
##
```

3 Limpieza de los datos.

Analizamos los valores nulos y vacíos. Para ello nos valemos de la salida de la función *df_status*, que nos muestra un resumen del estado de nuestras 8 variables actuales:

```
df_status(data)
```

```
##       variable q_zeros p_zeros q_na  p_na q_inf p_inf      type unique
## 1    Survived     549   61.62    0  0.00     0     0    factor      2
## 2      Pclass       0    0.00    0  0.00     0     0    factor      3
## 3         Sex       0    0.00    0  0.00     0     0    factor      2
## 4         Age       0    0.00  177 19.87     0     0   numeric     88
## 5       SibSp     608   68.24    0  0.00     0     0   integer      7
## 6       Parch     678   76.09    0  0.00     0     0   integer      7
## 7        Fare      15    1.68    0  0.00     0     0   numeric    248
## 8       Cabin       0    0.00  687 77.10     0     0 character    147
## 9    Embarked       0    0.00    2  0.22     0     0    factor      3
## 10      Price      15    1.68    0  0.00     0     0   numeric    248
## 11 FamilySize       0    0.00    0  0.00     0     0   numeric      9
```

3.1 Elementos nulos y ceros.

3.1.1 Elementos nulos

Procedemos a analizar los elementos nulos que se encuentran en el dataset

```
data %>% group_by(Embarked) %>% count(Embarked)
```

Embarked	n
C	168
Q	77
S	644
NA	2

En la tabla se observa que el campo **Age** tiene un 19.87% de nulos, **Cabin** más de un 77% y **Embarked** tiene dos nulos. Analizamos cada uno de los atributos para decidir la mejor decisión con respecto a los nulos.

3.1.1.1 Embarked Solo hay 2 valores nulos, al ser muy pocos casos y además la variable solo puede tomar 3 posibles valores, decidimos imputar el valor que más se repite, que es Embarked=S (Southampton)

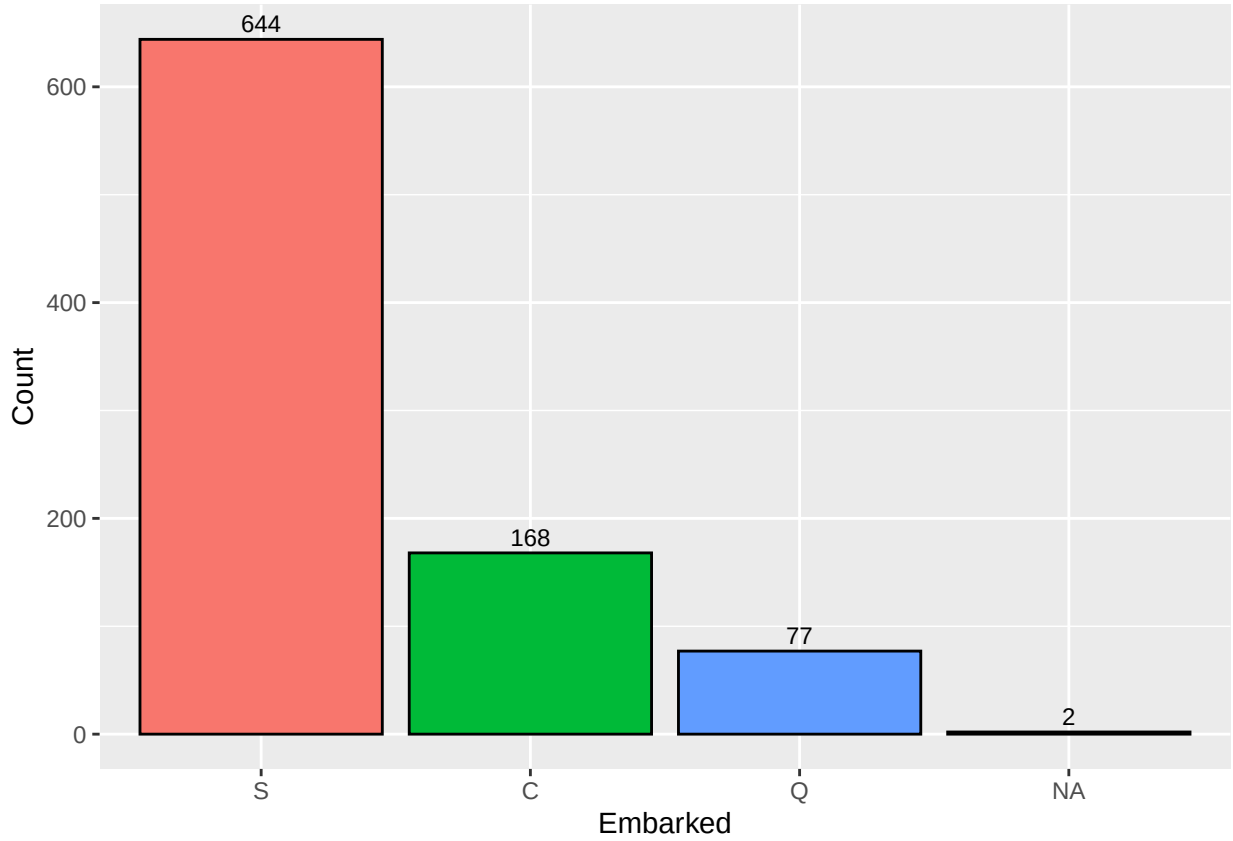
```
data %>% group_by(Embarked) %>% count(Embarked)
```

Embarked	n
C	168
Q	77
S	644
NA	2

```
data_Embarked <- sort(table(data$Embarked, useNA = "ifany"), decreasing = TRUE)
```

Gráficamente:

```
dat_plot <- as.data.frame(data_Embarked)
ggplot(dat_plot, aes(x=Var1, y=Freq, fill=Var1)) +
  geom_bar(stat = "identity", color = "black") +
  geom_text(aes(label=Freq), vjust = -0.4, color="black", size=3) +
  labs(x='Embarked', y='Count') +
  theme(legend.position = "none")
```



Asignamos el valor mas frecuente a los casos nulos:

```
data$Embarked[is.na(data$Embarked)] <- names(data_Embarked[1])
```

3.1.1.2 Cabin Dado que el número de nulos es muy elevado, un 77%, se opta por eliminar esta variable ya que tendríamos que imputar valores a una parte muy importante del dataset (con su consiguiente error). Además no parece que pueda ser una variable determinante para predecir la supervivencia.

```
data <- select(data, -Cabin)
```

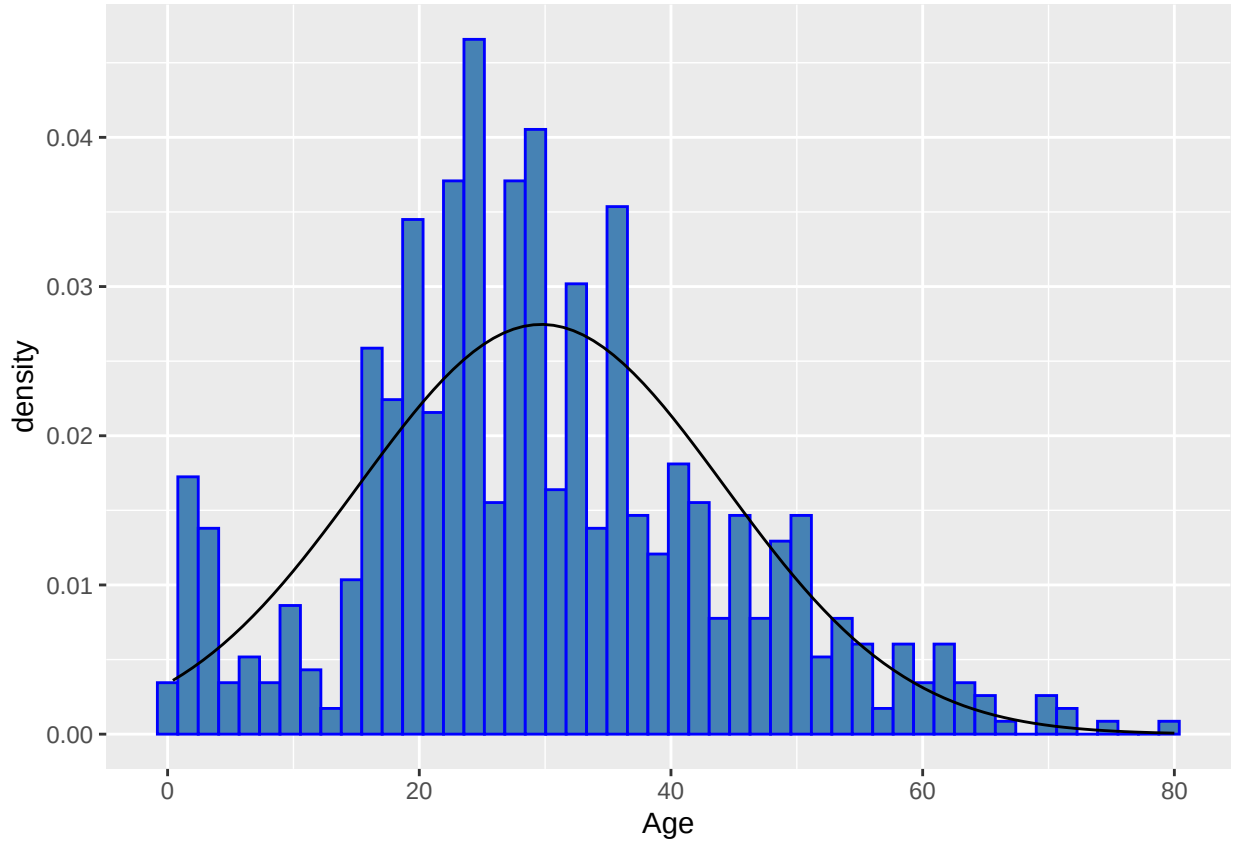
3.1.1.3 Age Tiene mas del un 19% de valores nulos (concretamente 177 registros)

Una opción para solucionarlos sería calcular la media del resto de registros e imputarla, pero vamos a estudiar si podemos delimitar la media a aplicar para obtener un resultado más depurado.

Primero analizamos la distribución de la variable **Age** teniendo en cuenta sólo los registros donde hay valores (es decir algo mas del 80% del dataframe). Para ello creamos un nuevo dataset sin los registros nulos de Age:

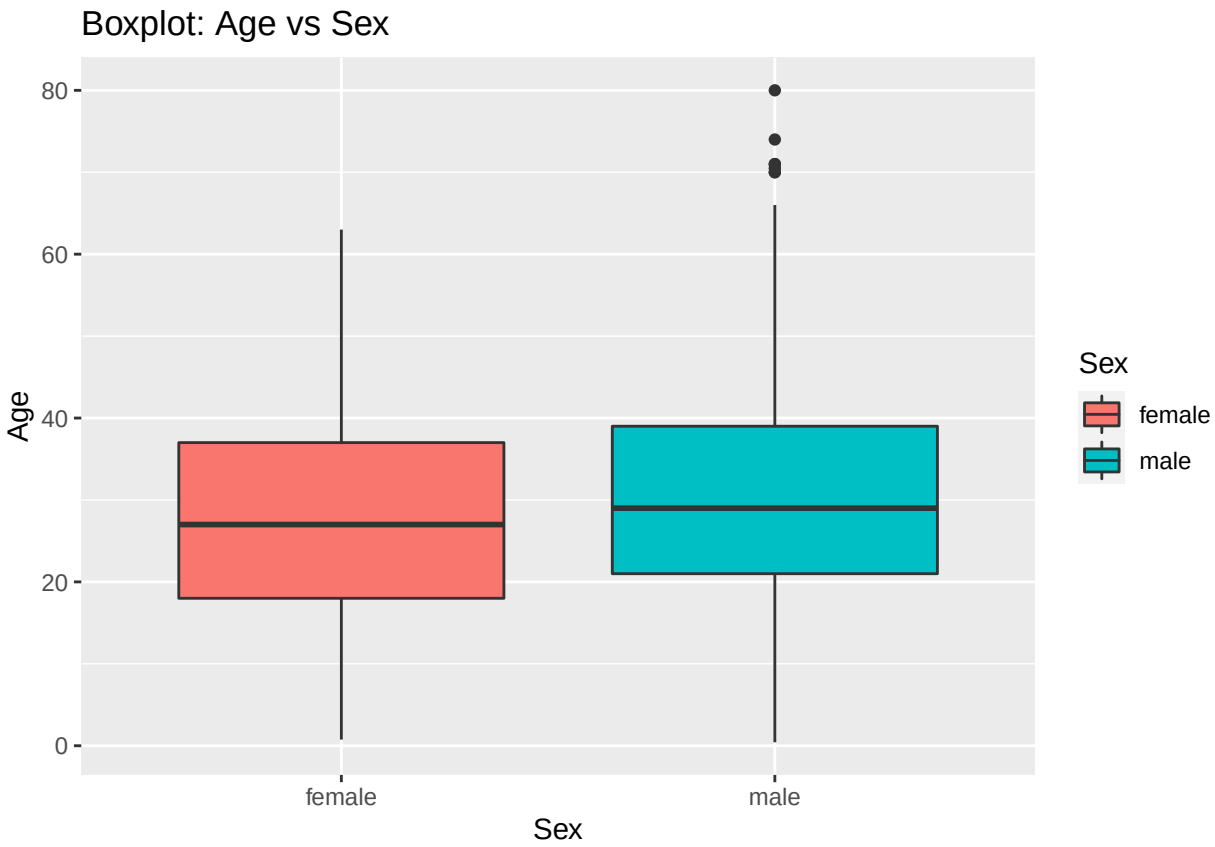
```
data_NoNA <- data[which(!is.na(data$Age)),]
```

Observamos gráficamente como se distribuye la variable Age en este dataset:



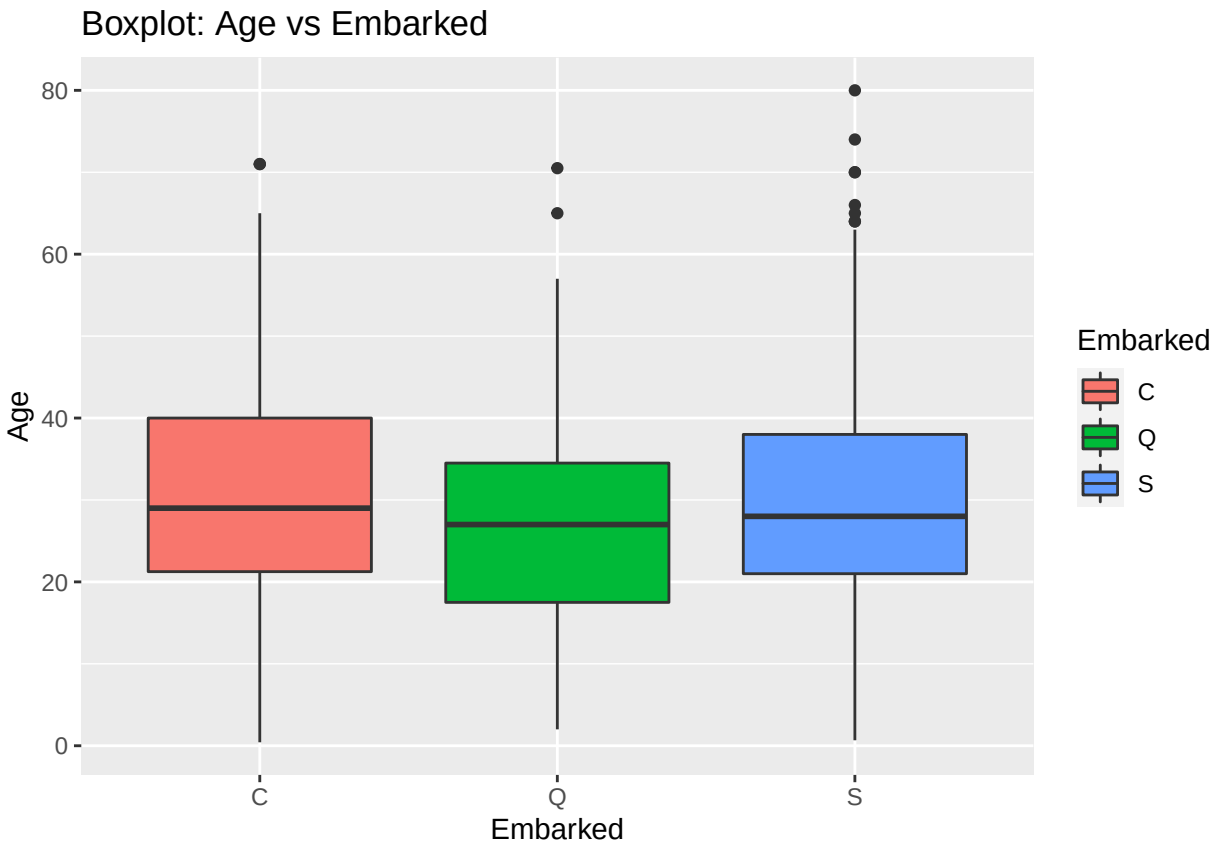
Comprobamos si hay alguna relación entre la edad y alguna otra variable del dataset para tenerlo en cuenta a la hora de imputar valores. Para ello iremos analizando Age con las distintas variables

3.1.1.3.1 Age vs Sex Observamos gráficamente la relación entre la edad y el género



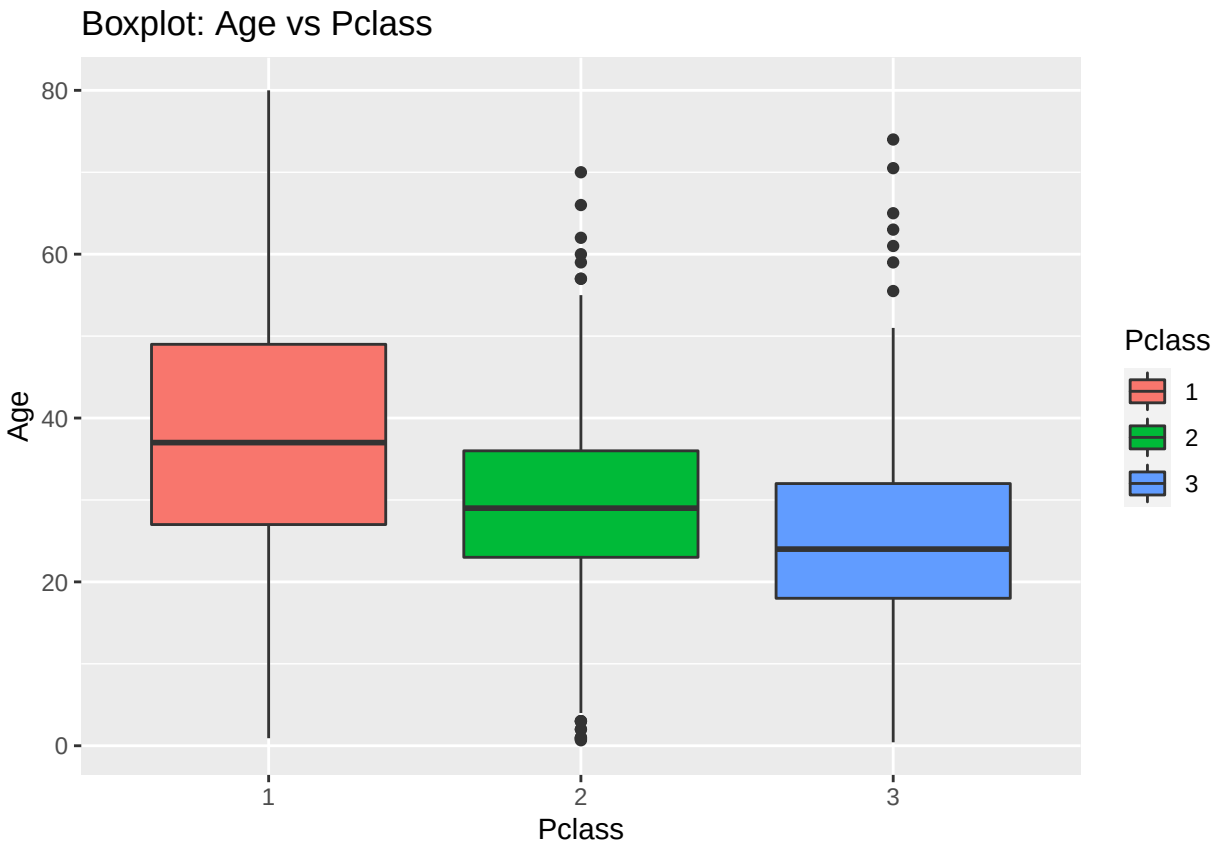
No se aprecian casi diferencias de edad en función del género

3.1.1.3.2 Age vs Embarked Comprobamos gráficamente la relación entre la edad y el puerto de Embarque



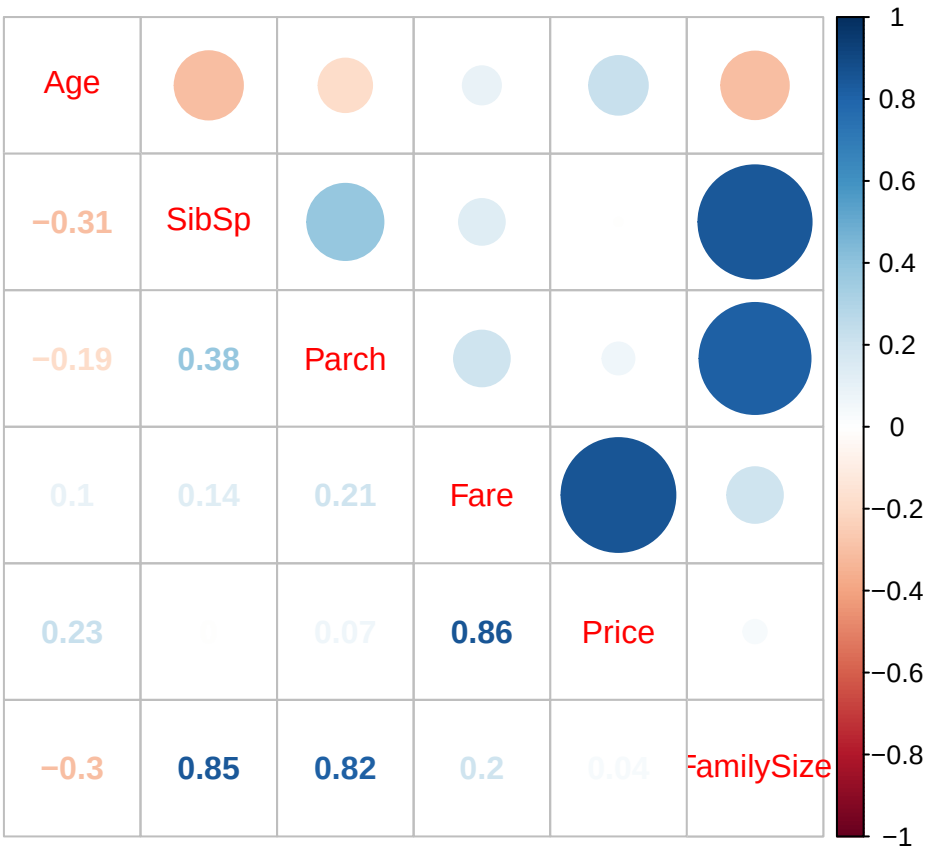
Tampoco se aprecian diferencias significativas en este caso con respecto a las medias de edad en cada una de las clases de **Embarked**.

3.1.1.3.3 Age vs Pclass Representamos visualmente la relación entre Age y Pclass:



En este caso si se observa una relación entre la edad y la clase en que viajaban los pasajeros: Los pasajeros de clase 1 (alta) tenían generalmente mayor edad que los de clase 2 (media) e igualmente sucede con los de clase 3 (baja).

3.1.1.3.4 Age vs Variables numéricas Vemos las distintas correlaciones de las variables numéricas

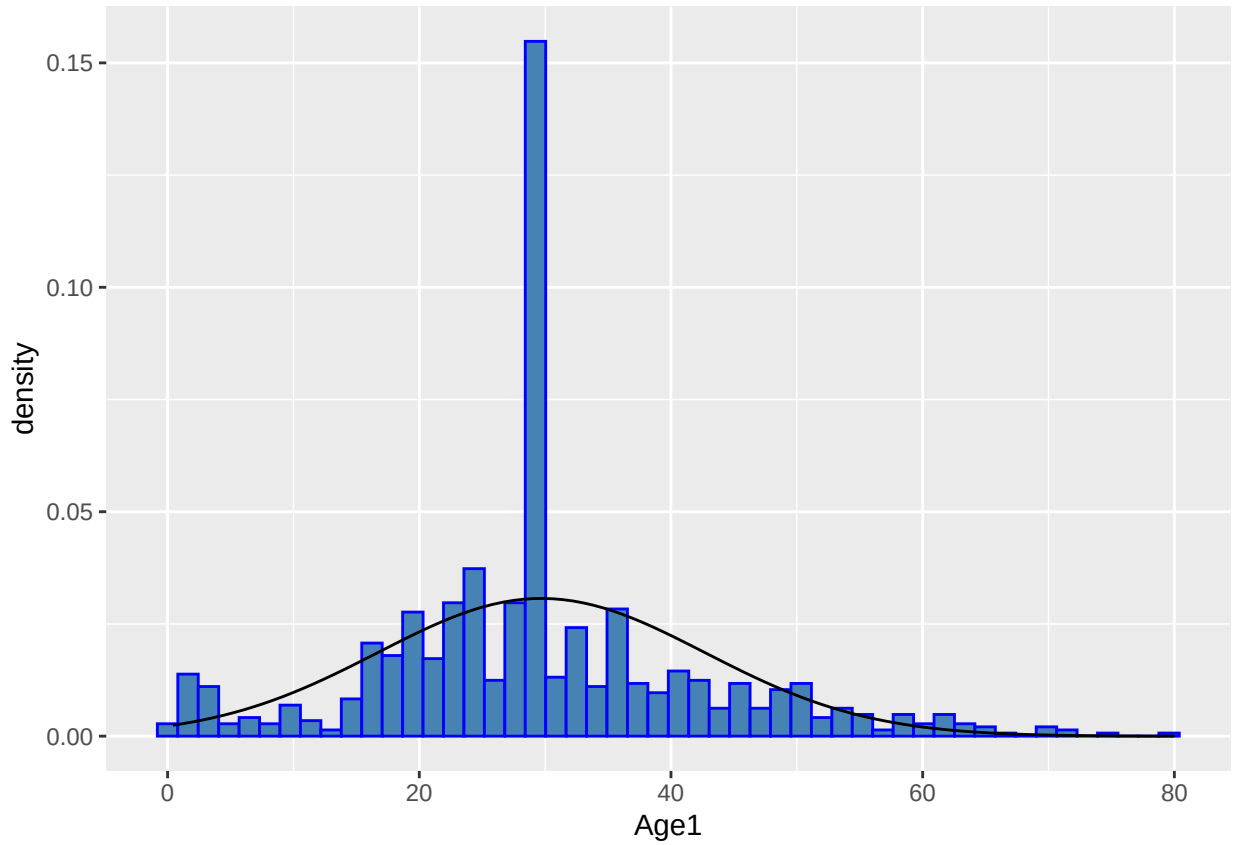


Se observa que existe correlación negativa con **SibSp** y con **Parch** (también con **FamilySize**, pero eso es completamente normal porque **FamilySize** es una transformación lineal de las otras 2). También hay correlación positiva con **Price**, pero entendemos que podemos tener en cuenta la parte familiar por el tema de esposa, hijos, hermanos, etc. puede tener efecto en la edad.

3.1.1.3.5 Imputación de valores a Age Tras el análisis de las diversas relaciones entre las variables y **Age** vamos a realizar cuatro simulaciones diferentes de imputación y nos quedaremos con una sola:

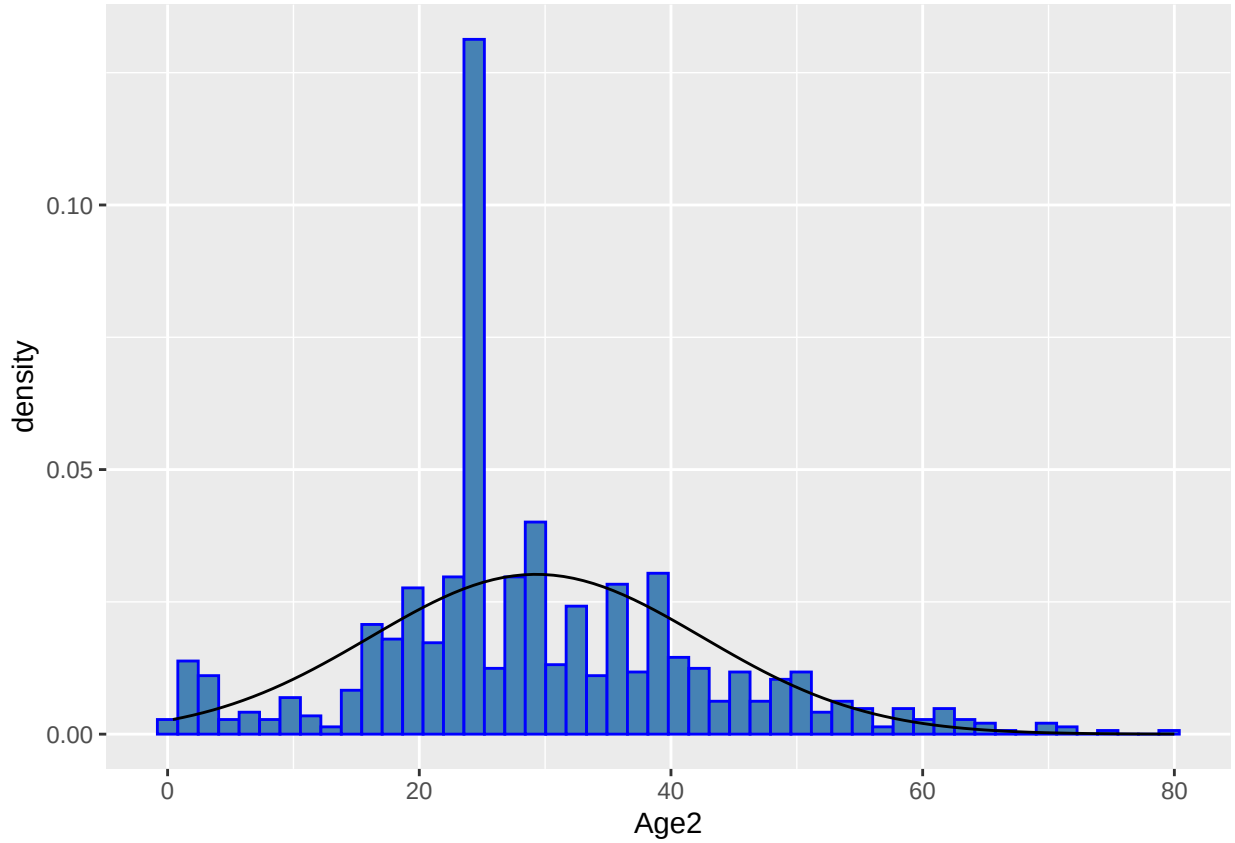
Caso 1: Imputar la media de Age** a todos los elementos faltantes**

```
data$Age1 <- data$Age
data$Age1[is.na(data$Age1)] <- mean(data_NoNA$Age)

ggplot(data, aes(Age1)) +
  geom_histogram(aes(y = ..density..), bins=50, fill="steelblue", color="blue") +
  stat_function(fun = dnorm, args = list(mean = mean(data$Age1), sd = sd(data$Age1)))
```



Caso 2: Imputar la media de Age** pero por cada una de las clases(**Pclass**) ya que hemos visto que hay una relación entre ambas variables.**

```
data$Age2 <- data$Age
data$Age2[is.na(data$Age2)&data$Pclass == 1] <- mean(data$Age2[!is.na(data$Age2)&data$Pclass == 1])
data$Age2[is.na(data$Age2)&data$Pclass == 2] <- mean(data$Age2[!is.na(data$Age2)&data$Pclass == 2])
data$Age2[is.na(data$Age2)&data$Pclass == 3] <- mean(data$Age2[!is.na(data$Age2)&data$Pclass == 3])

ggplot(data, aes(Age2)) +
  geom_histogram(aes(y = ..density..), bins=50, fill="steelblue", color="blue") +
  stat_function(fun = dnorm, args = list(mean = mean(data$Age2), sd = sd(data$Age2)))
```



Caso 3: Imputar la datos en Age, pero teniendo en cuenta Pclass, Parch y SibSp

Este caso surge al haber visto correlación entre las variables. Para este caso lo que hacemos es tener en cuenta las 3 variables y generar una agrupación de datos calculando la media para las combinaciones (recordemos que son variables discretas). Una vez que obtenemos esos valores medios, los imputamos. Luego verificamos si quedó algún valor sin imputar (que serán muy pocos) y para esos pocos casos faltantes, imputar los valores por cada clase.

```
medias_clase_fam <- as.data.frame(data_NoNA %>% group_by(Pclass, SibSp, Parch) %>% dplyr::summarize(Medi
```

```
## `summarise()` regrouping output by 'Pclass', 'SibSp' (override with `.groups` argument)
```

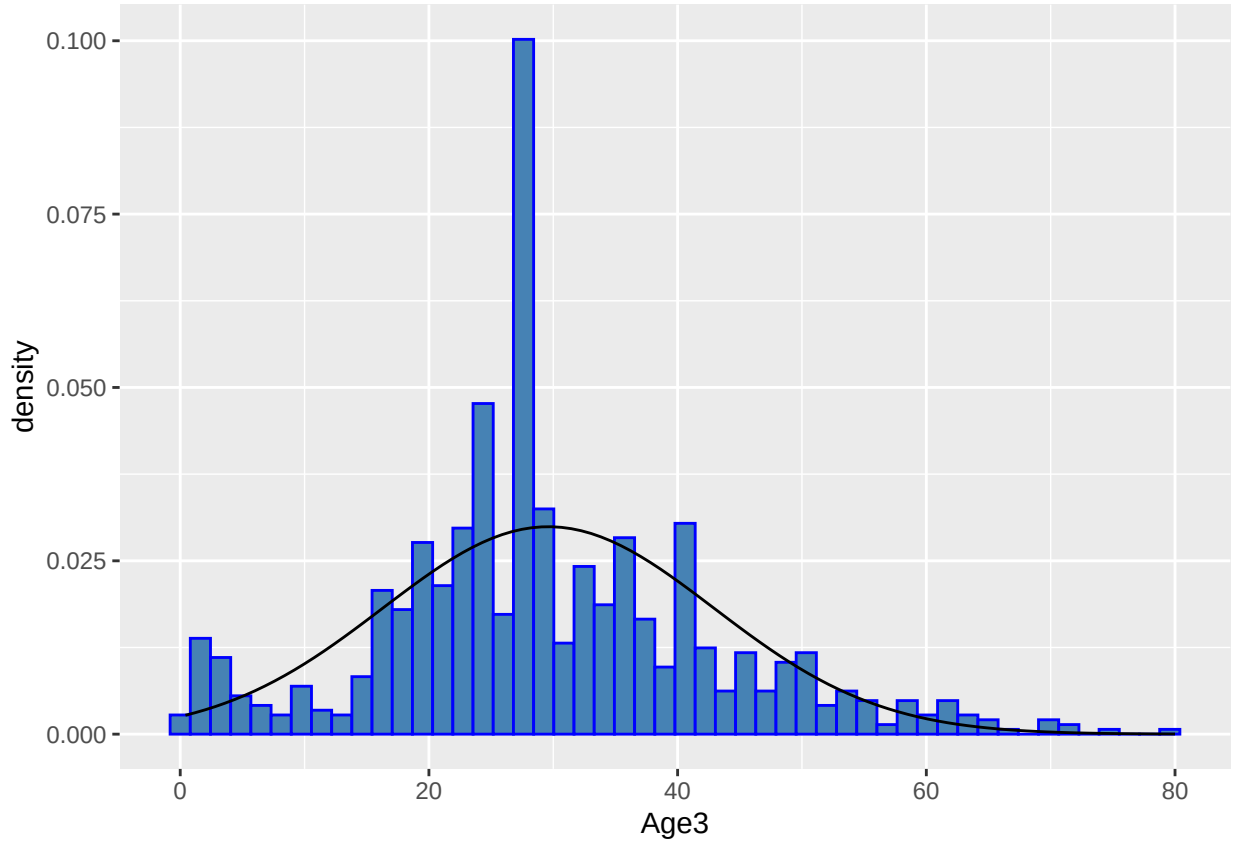
Hacemos un merge entre nuestro dataset original y el de Medias (por **Pclass**, **SibSp** y **Parch**) a través de las columnas usadas en la agrupación. El merge es un LEFT JOIN ya que puede que no existan todas las combinaciones en medias_clase_fam

```
data$Age3 <- data$Age
data <- merge(data, medias_clase_fam, by = c("Pclass", "SibSp", "Parch"), all.x = TRUE)
data$Age3[is.na(data$Age3)] <- data$Media_clase_fam[is.na(data$Age3)]
```

Como es posible que nos hayan quedado alguno sin poder asignar (al no existir la combinación Pclass + sibSp + Parch) a los que faltan (que son 7) les asignamos directamente por la **Pclass** sin tener en cuenta los otros valores

```
data$Age3[is.na(data$Age3)&data$Pclass == 1] <- mean(data$Age3[!is.na(data$Age3)&data$Pclass == 1])
data$Age3[is.na(data$Age3)&data$Pclass == 2] <- mean(data$Age3[!is.na(data$Age3)&data$Pclass == 2])
data$Age3[is.na(data$Age3)&data$Pclass == 3] <- mean(data$Age3[!is.na(data$Age3)&data$Pclass == 3])
```

Vemos ahora la distribución que nos queda



Caso 4: Imputando datos de Age**, con MICE (Multivariate Imputation via Chained Equations)**

En este caso se predicen los valores de **Age**, con el resto de valores observados (usamos para este caso **Pclass**, **SibSp**, **Parch**, **Sex** y **Age**).

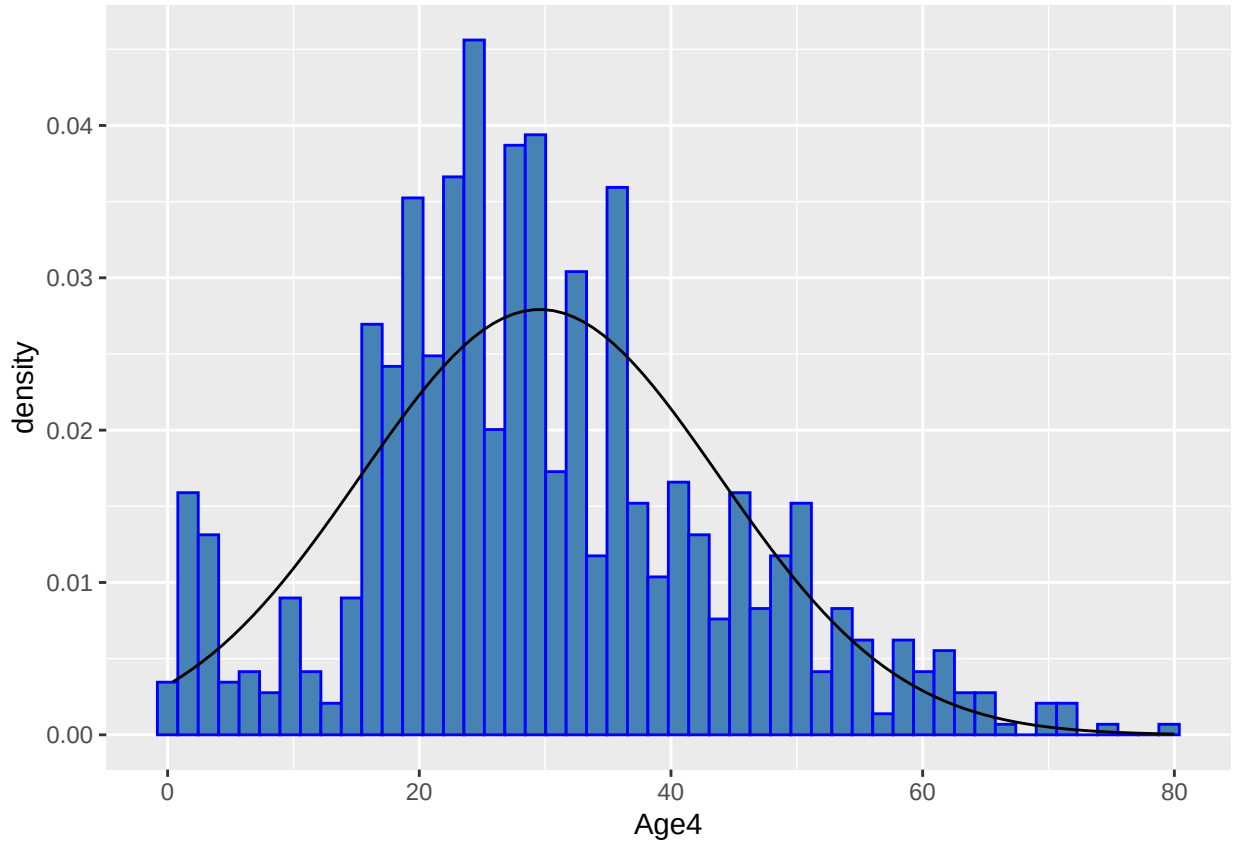
```
columnas <- c('Pclass', 'SibSp', 'Parch', 'Sex', 'Age')

mice_imputar <- mice(data = data[, columnas], method = "rf")
```

```
##
##  iter imp variable
##    1   1  Age
##    1   2  Age
##    1   3  Age
##    1   4  Age
##    1   5  Age
##    2   1  Age
##    2   2  Age
##    2   3  Age
##    2   4  Age
##    2   5  Age
##    3   1  Age
##    3   2  Age
##    3   3  Age
##    3   4  Age
##    3   5  Age
##    4   1  Age
```

```
##    4    2  Age
##    4    3  Age
##    4    4  Age
##    4    5  Age
##    5    1  Age
##    5    2  Age
##    5    3  Age
##    5    4  Age
##    5    5  Age
```

```
mice_completo <- mice::complete(mice_imputar)
data$Age4 <- data$Age
data$Age4[is.na(data$Age4)]<- mice_completo$Age[is.na(data$Age4)]
```

Tras la imputación observamos la gráfica:



Resumiendo el resultado de las 4 opciones de imputación mas la original:

```
summary(data$Age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    0.42   20.12   28.00   29.70   38.00   80.00     177
```

```
summary(data$Age1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.42   22.00   29.70   29.70   35.00   80.00
```

```
summary(data$Age2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.42   22.00   26.00   29.29   37.00   80.00
```

```
summary(data$Age3)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.42   22.00   28.24   29.60   37.00   80.00
```

```
summary(data$Age4)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.42   20.50   28.00   29.58   38.00   80.00
```

Tras ver los resultados nos decantamos por la cuarta opción (Age4), ya que la distribución se parece mucho más a la original. Asignamos y limpiamos el dataset:

```
data$Age <- data$Age4
data <- select(data, -Age1, -Age2, -Age3, -Age4, -Media_clase_fam)
```

3.1.2 Ceros

Recordamos los valores con ceros del dataset

```
##      variable q_zeros p_zeros q_na p_na q_inf p_inf    type unique
## 1      Pclass       0    0.00    0    0     0     0  factor      3
## 2       SibSp     608   68.24    0    0     0     0 integer      7
## 3       Parch     678   76.09    0    0     0     0 integer      7
## 4    Survived     549   61.62    0    0     0     0  factor      2
## 5         Sex       0    0.00    0    0     0     0  factor      2
## 6         Age       0    0.00    0    0     0     0 numeric     88
## 7        Fare      15    1.68    0    0     0     0 numeric    248
## 8    Embarked       0    0.00    0    0     0     0  factor      3
## 9       Price      15    1.68    0    0     0     0 numeric    248
## 10 FamilySize       0    0.00    0    0     0     0 numeric      9
```

Tenemos un alto número de ceros en **SibSp** y **Parch**, pero son valores válidos dentro del rango para estas variables, pues cuentan el número de acompañantes (familiares) del pasajero. Hay también un porcentaje pequeño de ceros en **Fare** (tarifa), esto podría tener algún sentido (por ejemplo, tickets sin coste por ser un premio o un regalo) por lo que, en principio, vamos a dejar presentes estos ceros.

3.1.3 Conclusión limpieza nulos y ceros

Viendo nuevamente los resultados con **summary**, hemos eliminado los nulos y tenemos un nuevo valor de media y mediana para **Age**

```
summary(data)
```

```
##  Pclass      SibSp           Parch        Survived     Sex           Age
##  1:216   Min.   :0.000   Min.   :0.0000   0:549    female:314   Min.   : 0.42
##  2:184   1st Qu.:0.000   1st Qu.:0.0000   1:342    male  :577   1st Qu.:20.50
##  3:491   Median :0.000   Median :0.0000                         Median :28.00
##          Mean   :0.523   Mean   :0.3816                         Mean   :29.58
##          3rd Qu.:1.000   3rd Qu.:0.0000                         3rd Qu.:38.00
##          Max.   :8.000   Max.   :6.0000                         Max.   :80.00
##       Fare        Embarked     Price           FamilySize
##  Min.   :  0.00   C:168    Min.   :  0.000   Min.   : 1.000
```

```
##  1st Qu.:  7.91   Q: 77    1st Qu.:  7.763   1st Qu.: 1.000
##  Median : 14.45   S:646    Median :  8.850   Median : 1.000
##  Mean   : 32.20            Mean   : 17.789   Mean   : 1.905
##  3rd Qu.: 31.00            3rd Qu.: 24.288   3rd Qu.: 2.000
##  Max.   :512.33            Max.   :221.779   Max.   :11.000
```

Y con "df_status" vemos también como nos han quedado los datos después de eliminar variables, y de imputar valores en las variables a los que le faltaban algunos valores. El dataset queda libre de nulos, y con los ceros que hemos aceptado que tiene que mantener y que son valores aceptables.

```
df_status(data)
```

```
##      variable q_zeros p_zeros q_na p_na q_inf p_inf    type unique
## 1      Pclass       0    0.00    0    0     0     0  factor      3
## 2       SibSp     608   68.24    0    0     0     0 integer      7
## 3       Parch     678   76.09    0    0     0     0 integer      7
## 4    Survived     549   61.62    0    0     0     0  factor      2
## 5         Sex       0    0.00    0    0     0     0  factor      2
## 6         Age       0    0.00    0    0     0     0 numeric     88
## 7        Fare      15    1.68    0    0     0     0 numeric    248
## 8    Embarked       0    0.00    0    0     0     0  factor      3
## 9       Price      15    1.68    0    0     0     0 numeric    248
## 10 FamilySize       0    0.00    0    0     0     0 numeric      9
```

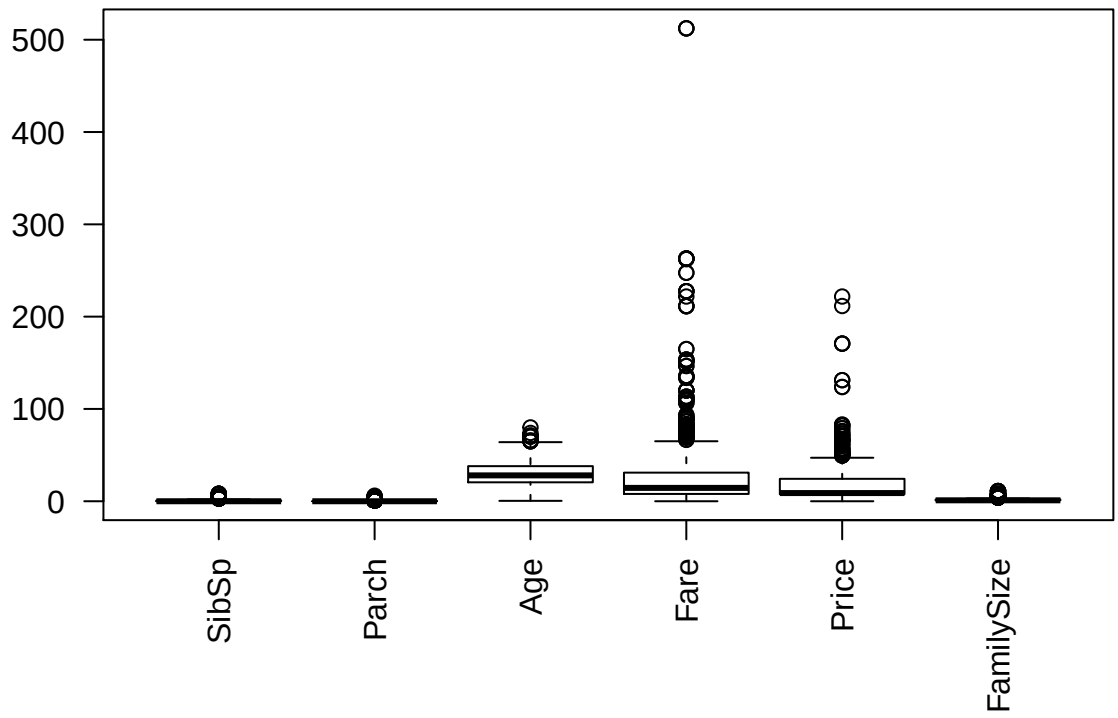
3.2 Identificación y tratamiento de valores extremos.

Se considera un valor extremo, outlier, a un valor fuera de rango. Son valores que se salen de la escala esperada visualizando el resto de las observaciones. En la actualidad, el criterio más habitual es considerar un valor extremo a aquel que se encuentra alejado de la media unas tres veces la desviación típica.

Este análisis solo tiene sentido realizarlo sobre las variables numéricas por lo que vamos a comenzar por separar aquellas variables que son numéricas y verlas en un gráfico bloxplot.

Gráfico boxplot variables numéricas

```
lista<-sapply(data, is.numeric)
data_num<-data[,lista]
boxplot(data_num,las=2)
```




```
var.continuas <- vector()
```

De todas las variables numéricas sólo 3 son contínuas: **Age**, **Price** y **Fare**

Analicemos esas tres variables por separado

3.2.1 Fare

El boxplot parece indicar que Fare tiene valores extremos. Los identificamos usando el criterio de tres veces la desviación típica:

```
data_out <- as.data.frame(data$Fare)
data_out$outlier <- FALSE
for (i in 1:ncol(data_out) - 1){
  columna = data_out[, i]
  if (is.numeric(columna)) {
    media = mean(columna)
    desviacion = sd(columna)
    data_out$outlier = (columna> (media+3*desviacion) | columna<(media-3*desviacion))
  }
}
table(data_out$outlier)
```

```
##
## FALSE  TRUE
##   871    20
```

Con este criterio tenemos identificados 20 posibles *outliers*, observando mas detenidamente los valores que nos indica boxplot.stats y teniendo en cuenta que el máximo es 512, no parecen exagerados:

```
boxplot.stats(data$Fare)$out
```

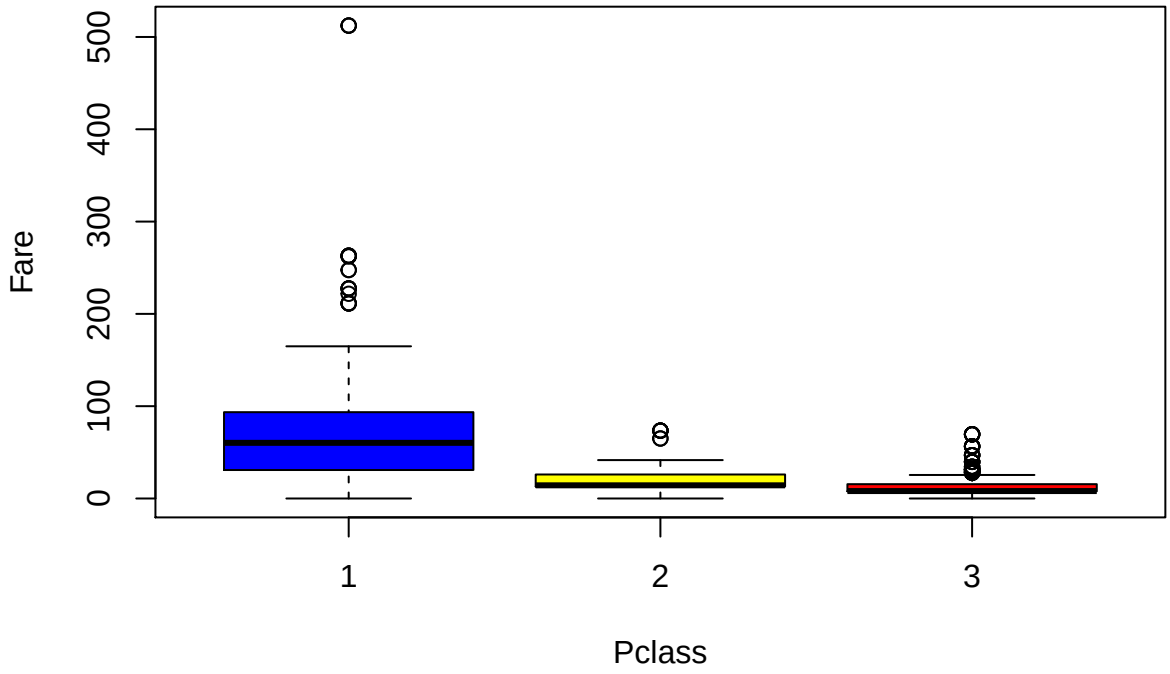
```
##   [1]  86.5000  86.5000  86.5000 151.5500 227.5250 227.5250 211.3375  80.0000
##   [9]  80.0000 135.6333  79.2000  79.2000  93.5000 512.3292  69.3000 221.7792
##  [17] 227.5250 134.5000 110.8833 135.6333  83.1583 135.6333 106.4250 153.4625
##  [25]  76.2917  78.8500  77.9583  69.3000  79.2000 146.5208 512.3292  76.7292
##  [33] 153.4625 153.4625  77.2875  77.2875 512.3292  83.1583 211.3375 247.5208
##  [41] 211.3375 247.5208 164.8667  71.0000  79.6500 113.2750 110.8833 211.5000
##  [49]  81.8583  76.7292  82.1708  90.0000  66.6000  89.1042  77.9583  90.0000
##  [57] 113.2750 113.2750  66.6000 106.4250  78.2667  83.4750  83.4750 133.6500
##  [65]  89.1042 227.5250  91.0792 108.9000  77.9583  90.0000  82.1708  75.2500
##  [73]  78.8500  71.2833 146.5208  76.7292  91.0792  78.2667 108.9000  79.6500
##  [81]  71.0000  79.6500 164.8667 110.8833 134.5000 110.8833  79.2000  83.1583
##  [89]  93.5000 120.0000 120.0000 120.0000 151.5500 151.5500 151.5500 120.0000
##  [97] 263.0000 133.6500  90.0000 262.3750 262.3750 263.0000 263.0000 263.0000
## [105]  73.5000  73.5000  73.5000  73.5000  73.5000  69.5500  69.5500  69.5500
## [113]  69.5500  69.5500  69.5500  69.5500
```

Los valores, además, están bien distribuidos, los más altos en las clases altas.

Por todo esto, decidimos no realizar ninguna acción con estos *outliers* ya que incluso pueden estar aportando información importante:

Gráfico outlier Fare

```
boxplot(data$Fare~data$Pclass,xlab="Pclass",ylab="Fare",
        col=c("blue","yellow","red"))
```



3.2.2 Price

Hacemos un análisis similar al realizado con Fare:

```
data_out <- as.data.frame(data$Price)
data_out$outlier <- FALSE
for (i in 1:ncol(data_out) - 1){
  columna = data_out[, i]
  if (is.numeric(columna)) {
    media = mean(columna)
    desviacion = sd(columna)
    data_out$outlier = (columna> (media+3*desviacion) | columna<(media-3*desviacion))
  }
}
table(data_out$outlier)
```

```
##
## FALSE  TRUE
##   878    13
```

De nuevo con boxplot.stats analizamos los outliers, el valor máximo 221:

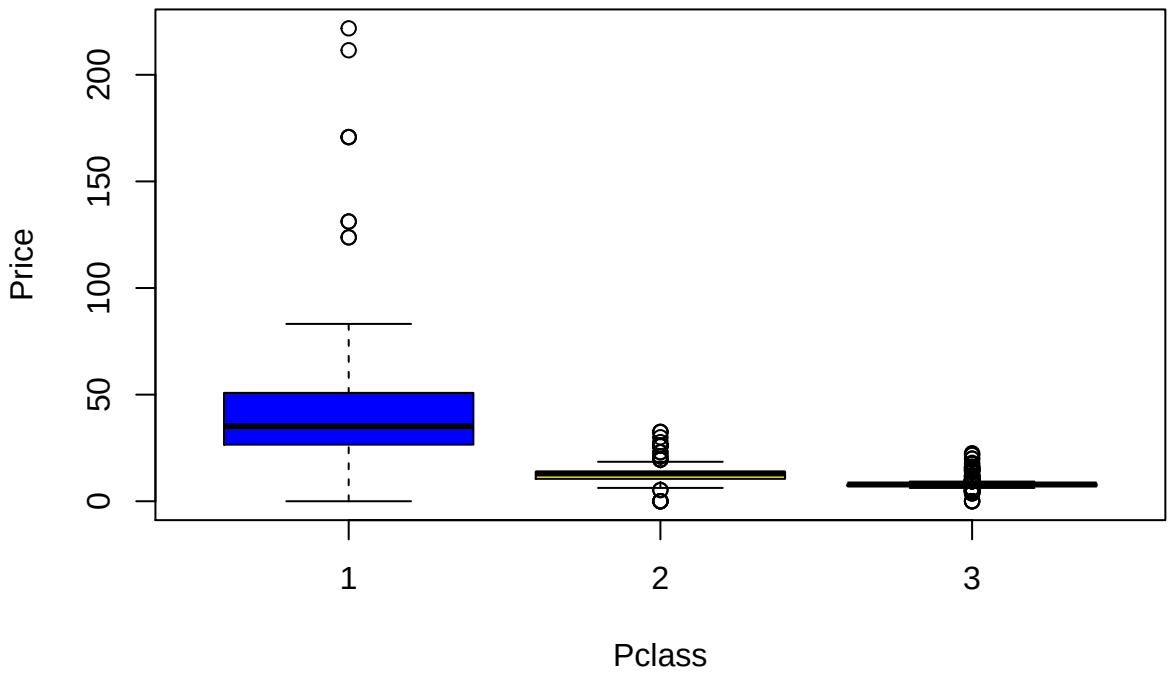
```
boxplot.stats(data$Price)$out
```

```
##  [1]  56.88125  56.88125  70.44583  50.00000  50.49580 170.77640  49.50420
##  [8] 221.77920  56.88125  67.25000  53.21250  51.86250  51.15417  76.29170
## [15]  79.20000  73.26040 170.77640  49.50420  51.15417  61.97920  51.15417
## [22] 170.77640  70.44583  61.37920 123.76040  70.44583 123.76040  63.35830
## [29]  82.43335  49.50000 211.50000  81.85830  51.86250  59.40000  53.21250
## [36]  66.82500  61.17500  53.10000  56.88125  54.45000  52.55420  52.00000
## [43]  55.44170  75.25000  71.28330  73.26040  54.45000  82.43335  67.25000
## [50]  79.20000  83.15830  65.75000  66.82500  51.47920 131.18750 131.18750
## [57]  65.75000  65.75000  65.75000
```

Y al igual que con **Fare**, los valores más altos en las clases altas:

Gráfico boxplot Price

```
boxplot(data$Price~data$Pclass,xlab="Pclass",ylab="Price",col=c("blue","yellow","red"))
```



Al igual que con **Fare** optamos por considerarlos valores válidos y no realizamos ninguna accion.

3.2.3 Age

De nuevo realizamos el mismo procedimiento que con las otras dos variables

```
data_out <- as.data.frame(data$Age)
data_out$outlier <- FALSE
for (i in 1:ncol(data_out) - 1){
  columna = data_out[, i]
  if (is.numeric(columna)) {
    media = mean(columna)
    desviacion = sd(columna)
    data_out$outlier = (columna> (media+3*desviacion) | columna<(media-3*desviacion))
  }
}
table(data_out$outlier)
```

```
##
## FALSE  TRUE
##   889     2
```

y el resultado de boxplot.stats

```
boxplot.stats(data$Age)$out
```

```
##  [1] 80.0 71.0 65.0 65.0 71.0 65.0 71.0 70.0 70.0 66.0 65.0 74.0 70.5
```

Donde vemos que podríamos tener 2 valores outliers, pero observando los valores devueltos por boxplot.stats

que son totalmente normales no los vamos a considerar outliers.

4 Análisis de los datos.

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

Del dataset completo nos interesa poder realizar diferentes análisis en función de diferentes subconjuntos de datos, como puede ser el género, la clase en la que viajan los pasajeros, el puerto en el que han embarcado, incluso se pueden definir grupos por edad, y ver realmente si es cierto y se cumplió aquello que dicen en las películas "las mujeres y los niños primero" y poder comprobar si efectivamente, los niños tienen mejor índice de supervivencia que los adultos. Podemos definir diferentes agrupaciones que usaremos más adelante para estudiar los casos por grupos.

4.1.1 Niños

Vamos a definir una variable Child para aquellos registros en los que la edad sea menor que 8

```
edad_corte = 8
data$Child[data$Age <= edad_corte] <- 1
data$Child[data$Age > edad_corte] <- 0
data$Child <- as.factor((data$Child))
```

4.1.2 Género

Agrupamos por el género, creando una variable para cada uno de los géneros

```
Mujeres <- data[which(data$Sex == 'female'),]
Hombres <- data[which(data$Sex == 'male'),]
```

4.1.3 Lugar de embarque

Agrupamos por el lugar de embarque creando una variable por cada uno de los lugares

```
EmbarqueC <- data[which(data$Embarked == 'C'),]
EmbarqueQ <- data[which(data$Embarked == 'Q'),]
EmbarqueS <- data[which(data$Embarked == 'S'),]
```

4.1.4 Clase

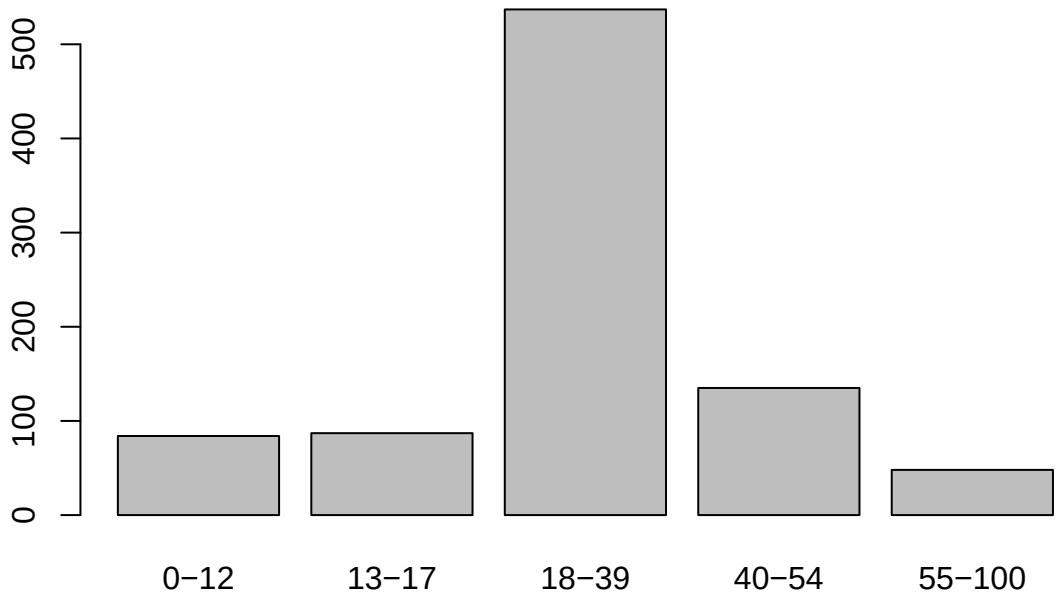
Agrupamos por clase

```
FirstClass <- data[which(data$Pclass == 1),]
SecondClass <- data[which(data$Pclass == 2),]
ThirdClass <- data[which(data$Pclass == 3),]
```

4.1.5 Edades

Vamos a discretizar los valores por grupos de edades, añadiendo una columna al dataset (AgeInterval)

```
##
##   0-12  13-17  18-39  40-54 55-100
##     84     87    537    135     48
```



4.2 Comprobación de la normalidad y homogeneidad de la varianza.

4.2.1 Normalidad

Vamos a verificar si las variables cuantitativas continuas siguen una distribución normal. Algunos test estadísticos requieren que las variables que van a ser analizadas sigan una distribución normal, por tanto tenemos que conocer cuáles son las distribuciones de nuestras variables continuas. Las únicas variables cuantitativas continuas que tenemos en el dataset original son las variables **Age** y **Fare**. Además, hemos generado una nueva variable a partir de **Fare**, que hemos llamado **Price**, y que, por tanto también es una variable cualitativa continua.

En general, la prueba de *Shapiro-Wilk* se considera una prueba muy potente para contrastar la normalidad de distribuciones. Se asume como hipótesis nula que la población sigue una distribución normal. Si el p-valor obtenido es inferior al nivel de significancia (normalmente $= 0{,}05$) entonces se rechaza la hipótesis nula (y por tanto se concluye que los datos no vienen de una distribución normal). En cambio, si el p-valor es superior al nivel de significancia, entonces no se puede rechazar la hipótesis nula y se asume que los datos siguen una distribución normal. Para poder tener más seguridad, vamos a aplicar otros dos métodos, la prueba de *Anderson-Darling* y la prueba de *Kolmogorov-Smirnov* (conocida también como K-S)

Creamos un data frame para resumir los tests:

```
tabla.normalidad <- data.frame('variable' = character(),
                               'Test de Normalidad' = character(),
                               'Valor Estadístico' = numeric(),
                               'p-value' = numeric(),
                               stringsAsFactors = FALSE)
str(tabla.normalidad)
```

```
## 'data.frame':    0 obs. of  4 variables:
##  $ variable          : chr
##  $ Test.de.Normalidad: chr
##  $ Valor.Estadístico : num
##  $ p.value           : num
```

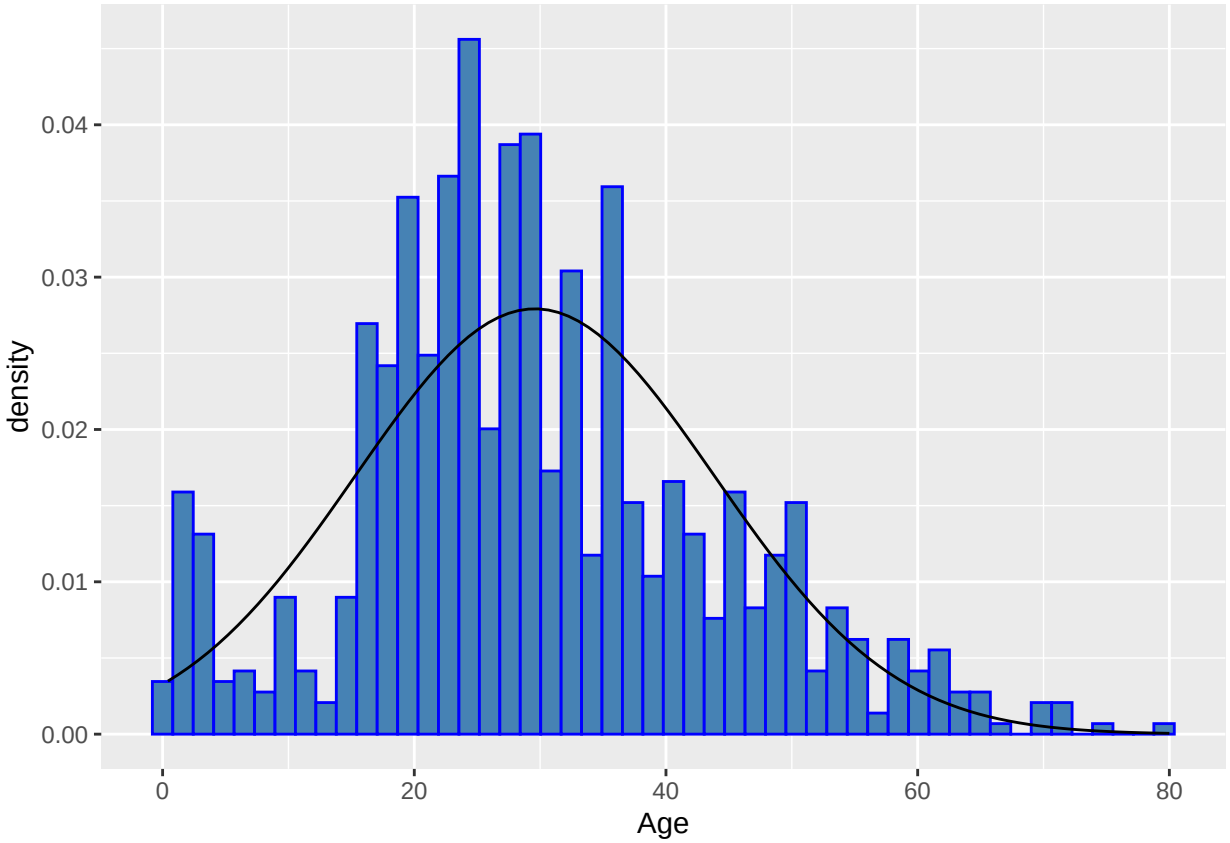
Ahora recorremos todas las variables continuas aplicando los tres tests y añadiéndolos al dataframe:

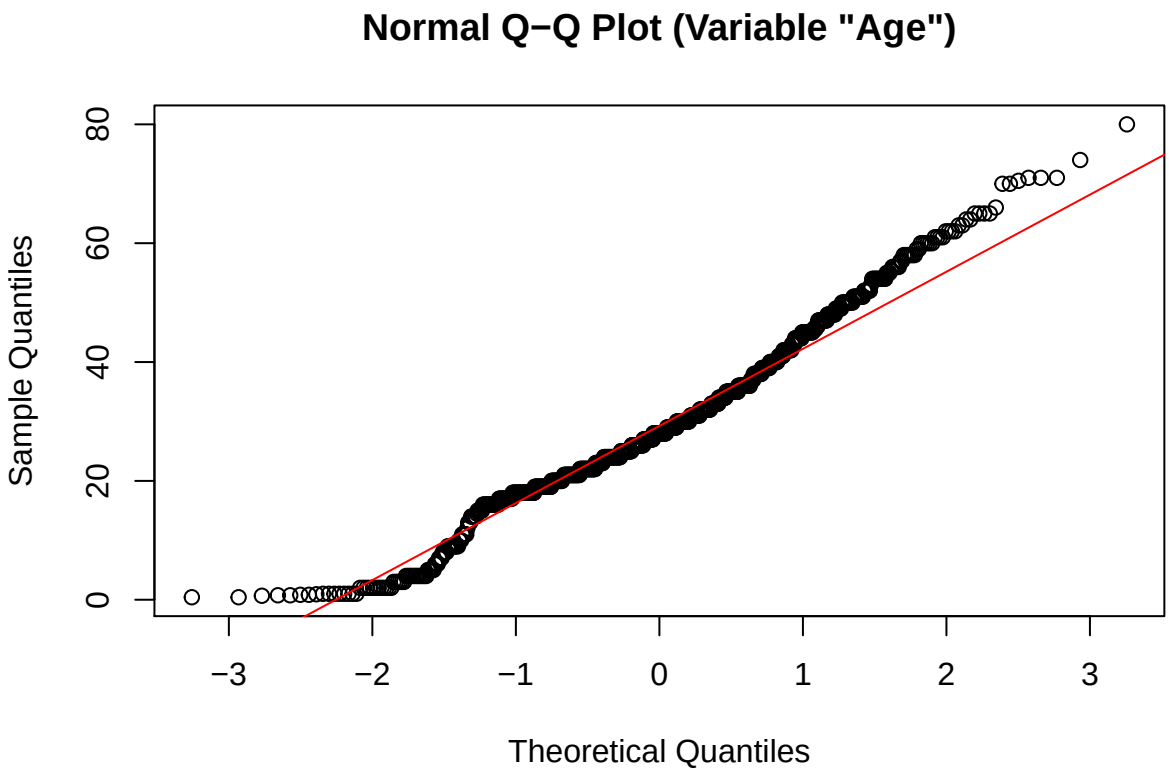
```
var.continuas <-c("Age", "Fare", "Price")

for (i in 1:length(var.continuas)){
  variable = var.continuas[i]
  #Test Shapiro-wil
  test = shapiro.test(data[,variable])
  tabla.normalidad[nrow(tabla.normalidad)+1,] = c(variable, test$method,
                                                  test$statistic, test$p.value)

  #Test Anderson-Darling
  test = ad.test(data[,variable])
  tabla.normalidad[nrow(tabla.normalidad)+1,] = c(variable, test$method,
                                                  test$statistic, test$p.value)

  #Test Kolmogorov-Smirnov
  test = ks.test(data[,variable], "pnorm", mean=mean(data[,variable]),
                 sd=sd(data[,variable]))
  tabla.normalidad[nrow(tabla.normalidad)+1,] = c(variable, test$method,
                                                  test$statistic, test$p.value)
}

knitr::kable(tabla.normalidad)
```

variable	Test.de.Normalidad	Valor.Estadístico	p.value
Age	Shapiro-Wilk normality test	0.978939808095505	4.80383070207932e-10
Age	Anderson-Darling normality test	5.96070197972574	1.14279492155628e-14
Age	One-sample Kolmogorov-Smirnov test	0.0712614538303797	0.000234936344110315
Fare	Shapiro-Wilk normality test	0.521891302117355	1.08404452322613e-43
Fare	Anderson-Darling normality test	122.169627214592	3.7e-24
Fare	One-sample Kolmogorov-Smirnov test	0.281848040985975	0
Price	Shapiro-Wilk normality test	0.566484900244852	3.00259150054125e-42
Price	Anderson-Darling normality test	110.325199878766	3.7e-24
Price	One-sample Kolmogorov-Smirnov test	0.26829589819932	0

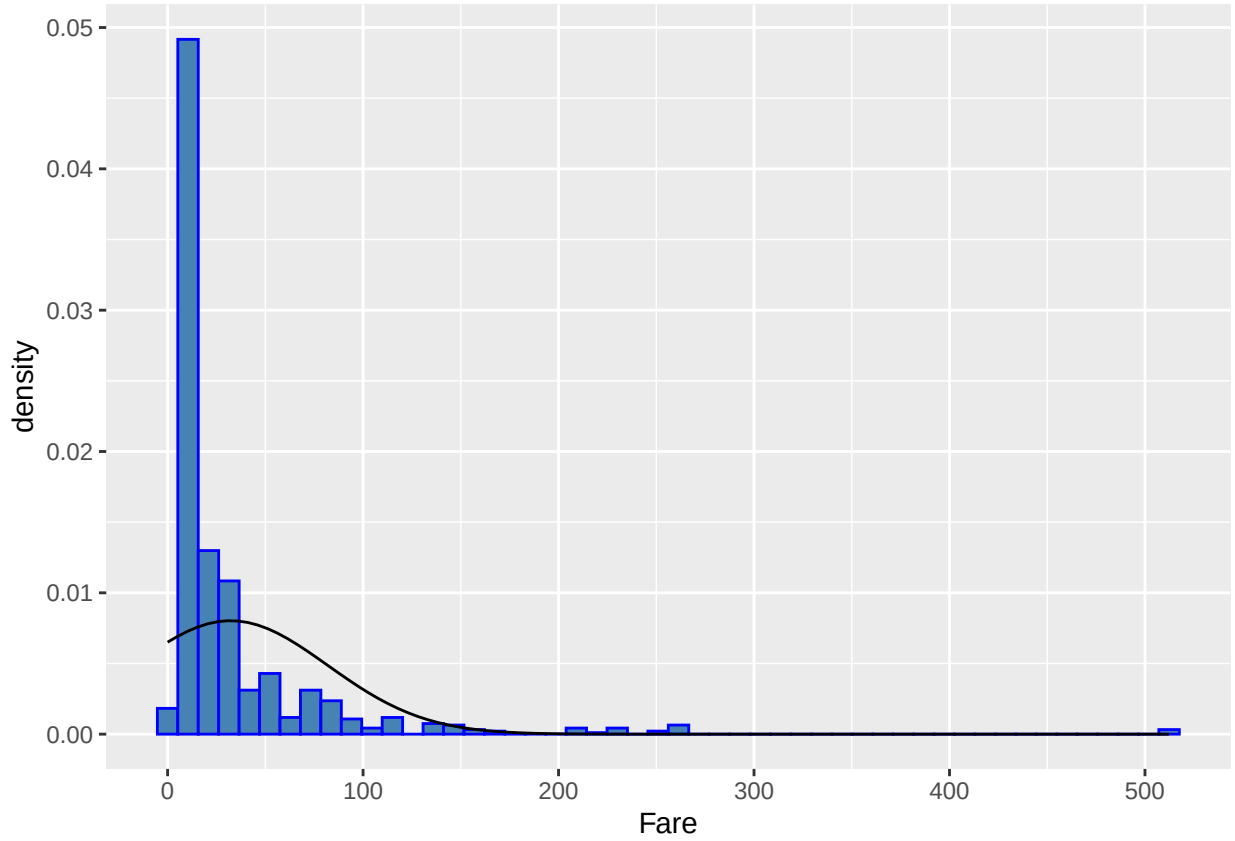
Con estos resultados se puede decir que ninguna de las 3 variables sigue una distribución normal, en todos los casos el *p-value* ha sido inferior a 0.05 y por tanto se han rechazado la hipótesis nula (que la variable sigue una distribución normal).

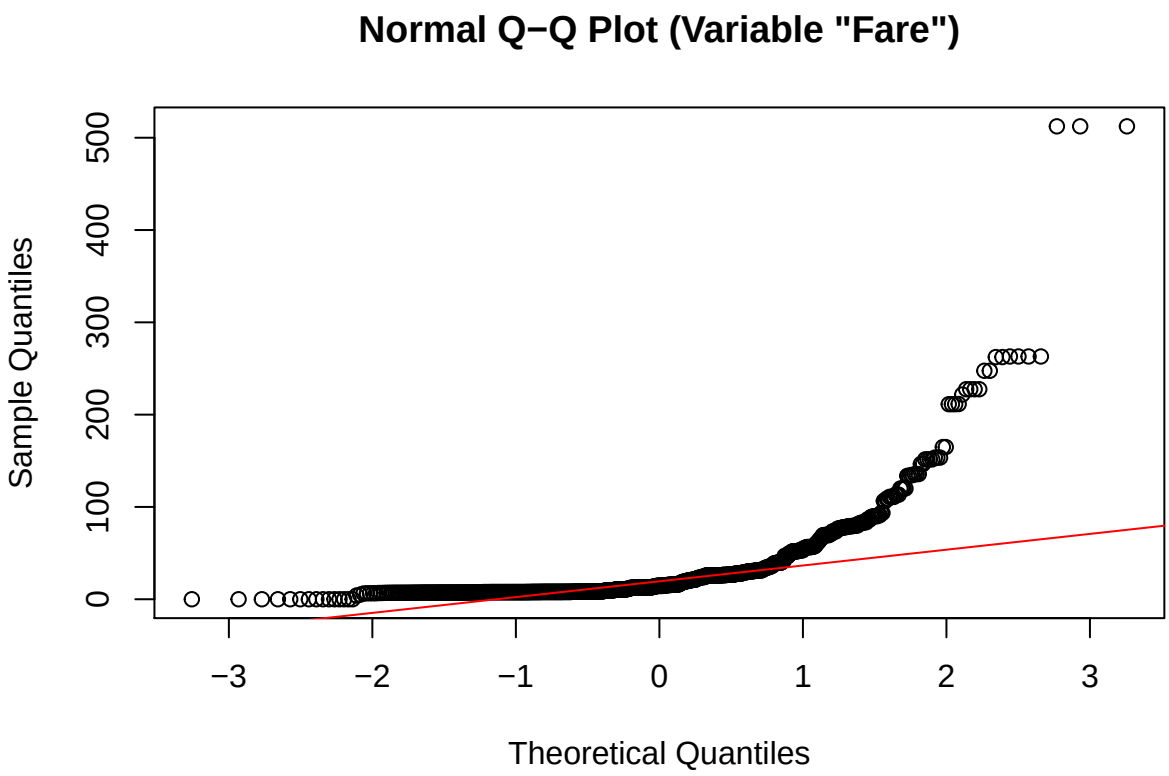
No obstante, vamos a revisar gráficamente la distribución de cada una de las variables usando su histograma, curva de densidad y gráficas Q-Q


density
0.04
0.03
0.02
0.01
0.00
0
20
40
60
80
Age

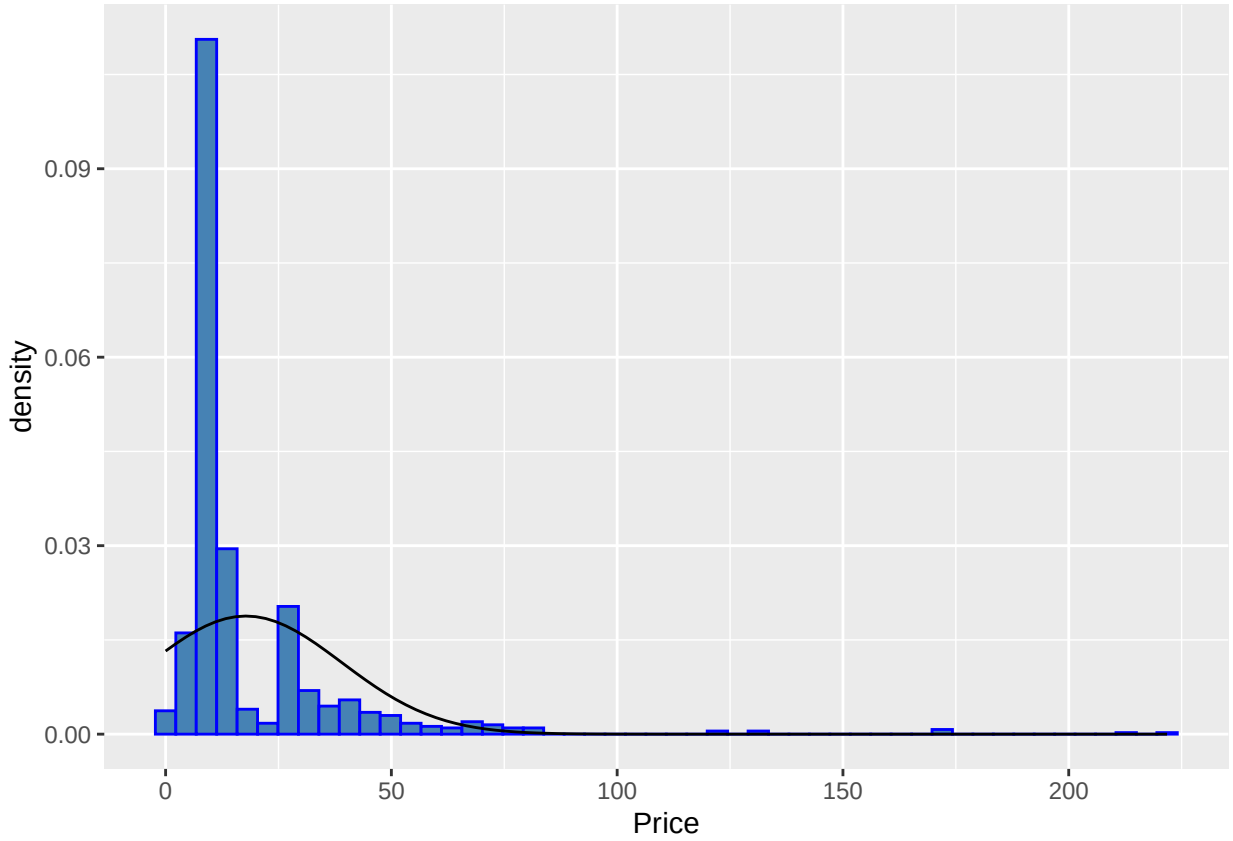

Normal Q−Q Plot (Variable "Age")
Sample Quantiles
Theoretical Quantiles
0
20
40
60
80
−3
−2
−1
0
1
2
3

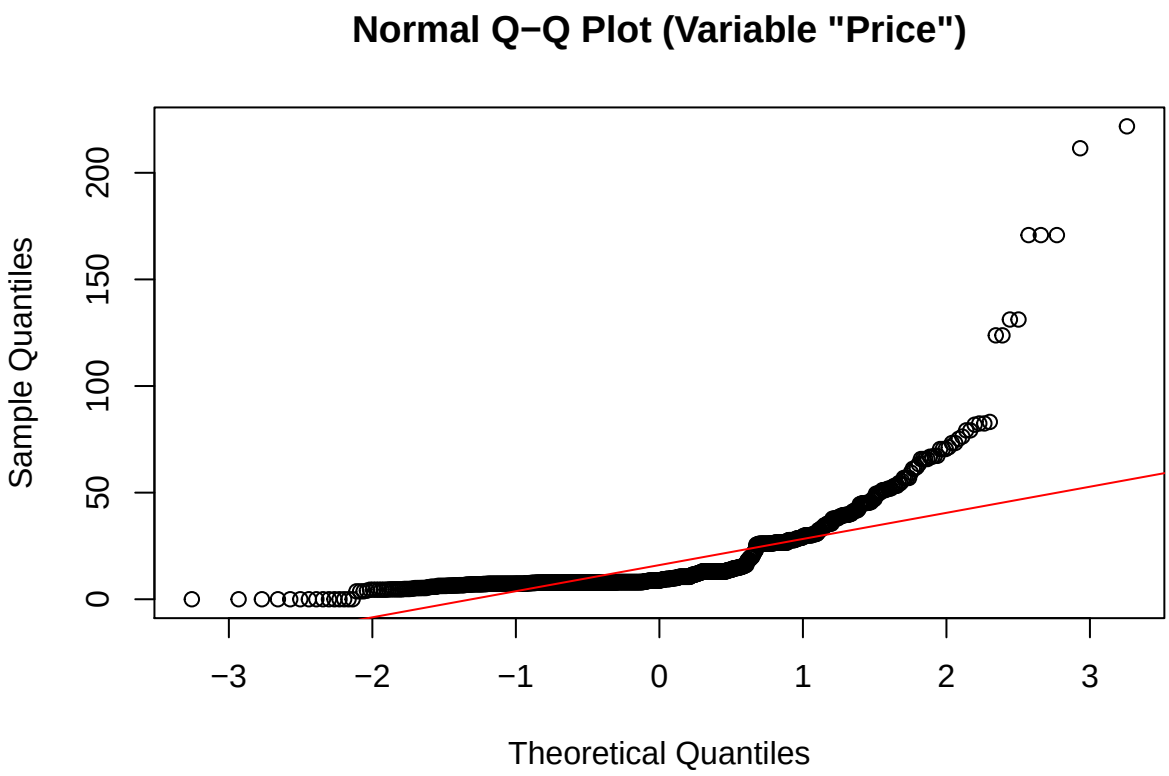
```
## [1] "Age"
```


0.05
0.04
0.03
0.02
0.01
0.00
density
0
100
200
300
400
500
Fare


Normal Q−Q Plot (Variable "Fare")
Sample Quantiles
0
100
200
300
400
500
−3
−2
−1
0
1
2
3
Theoretical Quantiles

```
## [1] "Fare"
```


density
0.09
0.06
0.03
0.00
0
50
100
150
200
Price




```
## [1] "Price"
```

TODO: ¿Conclusión del análisis gráfico?

4.2.2 Homocedasticidad (comprobación de varianzas)

Cuando comparamos varianzas lo que estamos comprobando es que las varianzas entre los grupos a comparar son iguales. Si los datos siguen una distribución normal, podemos usar el test de *Levene*, en caso contrario podemos usar por ejemplo el test de *Fligner-Killeen*, que es la alternativa no paramétrica que se utiliza cuando los datos no siguen una distribución normal (o cuando hay problemas con outliers no resueltos)

En ambos test (*Levene* y *Fligner-Killeen*), la hipótesis nula asume la igualdad de varianzas en los diferentes grupos de datos, con lo que si el p-value obtenido es inferior al nivel de significancia (generalmente alpha = 0,05) se rechaza la hipótesis nula y se concluye que hay heterocedasticidad.

4.2.2.1 Age

La variable **Age** está próxima a una distribución normal por lo cual podemos utilizar el test de *Levene* para la comprobación de varianzas.

4.2.2.1.1 Age con Survived

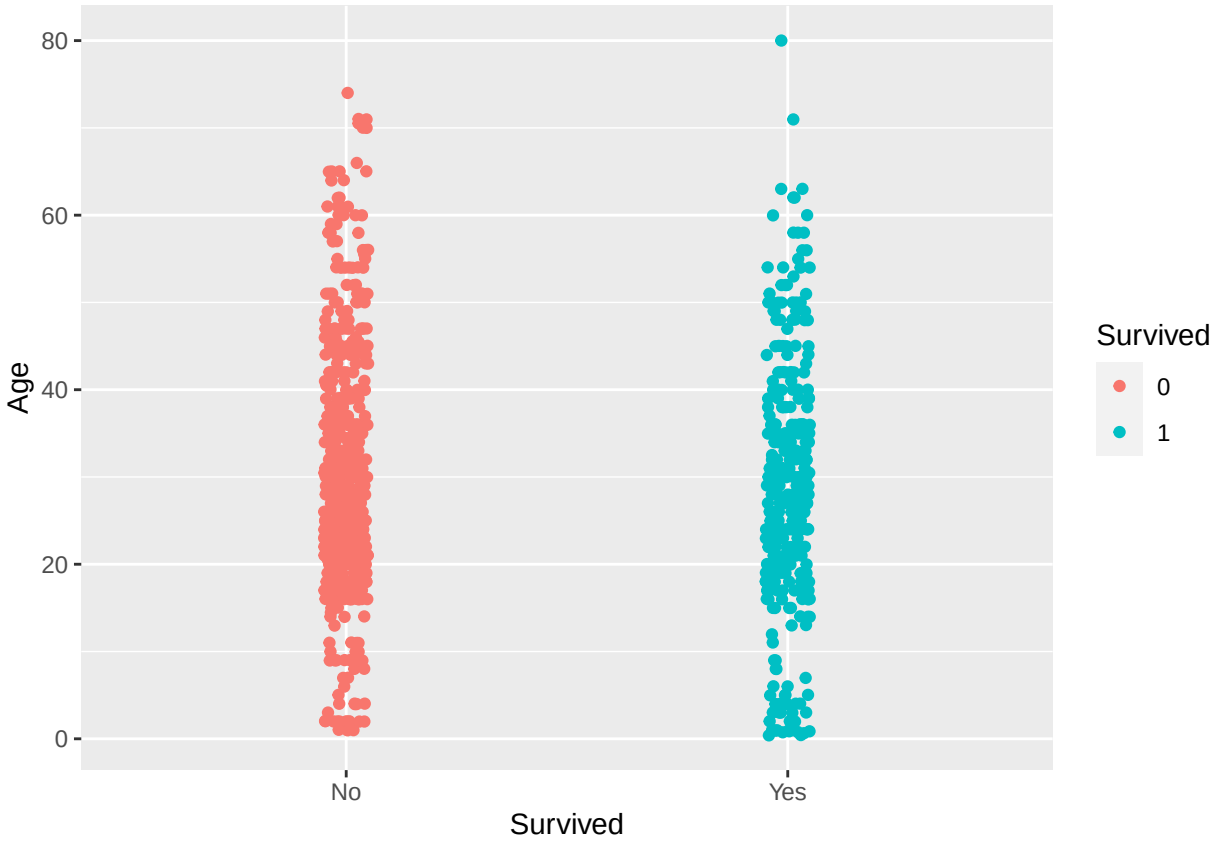
Primero analizamos si las varianzas son iguales cuando comprobamos **Age** y el grupo es **Survived**, es decir estamos comprobando la homogeneidad de varianzas de la edad en los grupos de supervivientes y no supervivientes.

```
leveneTest(data = data, Age ~ Survived, center = mean)
```

	Df	F value	Pr(>F)
group	1	0.0509905	0.8214004
	889	NA	NA

En este caso el p-value es superior a 0.05 y por tanto asumimos que hay **homogeneidad de varianzas** entre los grupos

Representación muestras forma gráfica




```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  Age by Survived
## Fligner-Killeen:med chi-squared = 0.1955, df = 1, p-value = 0.6584
```

4.2.2.1.2 Age con Embarked

Ahora Comprobamos también cómo se comportan las varianzas cuando se trata de la variable edad **Age** con **Embarked**.

```
leveneTest(data = data, Age ~ Embarked, center = mean)
```

	Df	F value	Pr(>F)
group	2	2.217208	0.1095154
	888	NA	NA

En este caso hay **homogeneidad de varianzas** de **Age** en los grupos que define la variable **Embarked**

4.2.2.1.3 Age con Pclass

Ahora Comprobamos también cómo se comportan las varianzas cuando se trata de la variable edad **Age** con **Pclass**. En este caso vamos a aplicar tanto el test de *Levene* como el de *Fligner-Killeen*:

```
leveneTest(data = data, Age ~ Pclass, center = mean)
```

	Df	F value	Pr(>F)
group	2	7.173114	0.0008122
	888	NA	NA

```
fligner.test(Age ~ Pclass, data = data)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  Age by Pclass
## Fligner-Killeen:med chi-squared = 13.782, df = 2, p-value = 0.001017
```

En los dos tests (paramétrico y no paramétrico), se rechaza la hipótesis nula, con lo que hay **heterogeneidad de varianzas** entre las muestras de **Age** cuando se agrupan por **Pclass**

4.2.2.1.4 Age con Sex Para concluir con el atributo **Age** realizamos tambien el análisis con respecto a **Sex**. De nuevo aplicamos *Levene* y *Fligner-Killen*

```
leveneTest(data = data, Age ~ Sex, center = mean)
```

	Df	F value	Pr(>F)
group	1	0.0369504	0.8476099
	889	NA	NA

```
fligner.test(Age ~ Sex, data = data)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  Age by Sex
## Fligner-Killeen:med chi-squared = 0.026265, df = 1, p-value = 0.8713
```

En los dos tests el p-value es claramente superior a 0.05 por lo que podemos afirmar que sí hay **homogeneidad de varianzas** para *Age* cuando está agrupado por *Sex*

4.2.2.1.5 Resumen Age Resumimos en una tabla los resultados de homogeneidad de la varianza de Age con respecto a los distintos atributos:

Age con respecto a	resultado
Survived	homogeneidad
Embarked	homogeneidad
Pclass	heterogeneidad
Sex	homogeneidad

4.2.2.2 Fare Hacemos el mismo estudio con la variable **Fare** pero ahora usaremos *Fligner*. Mostramos los test con **Sex**, **Survived**, **Embarked**, **Pclass**

```
fligner.test(Fare ~ Sex, data = data)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  Fare by Sex
## Fligner-Killeen:med chi-squared = 55.949, df = 1, p-value = 7.436e-14
```

```
fligner.test(Fare ~ Survived, data = data)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  Fare by Survived
## Fligner-Killeen:med chi-squared = 96.253, df = 1, p-value < 2.2e-16
```

```
fligner.test(Fare ~ Embarked, data = data)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  Fare by Embarked
## Fligner-Killeen:med chi-squared = 133.23, df = 2, p-value < 2.2e-16
```

```
fligner.test(Fare ~ Pclass, data = data)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  Fare by Pclass
## Fligner-Killeen:med chi-squared = 365.8, df = 2, p-value < 2.2e-16
```

Comprobamos la variable **Fare**, los resultados indican que hay **heterogeneidad de varianza** de esas variables respecto a los grupos con los que se ha aplicado el test no paramétrico.

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

4.3.1 Análisis de relaciones entre variables

Primero vamos a realizar diversos análisis de relaciones variables (principalmente pares de variables), para aclarar la dependencia entre ellas y como pueden afectar a las posibilidades de supervivencia

4.3.1.1 Age y Sex Primero vamos a analizar si hay diferencias significativas entre la media de edad de mujeres y de hombres. Para hacerlo aplicamos el test paramétrico *t-test de Student*, que requiere que las muestras a comparar sigan una distribución normal.

Ya vimos que **Age** no sigue exactamente una distribución normal, pero si consideramos el teorema central del límite con una muestra suficientemente grande (mayor de 30) se puede asumir que la variable sigue una distribución normal. Si la variable no siguiese una distribución normal podríamos aplicar una prueba no paramétrica como por ejemplo el test de *Mann-Whitney*.

Para ejecutar la prueba *t de Student*, disponemos en R del comando "t.test". Este mismo comando nos permite realizar el test de *Welch* cuando las varianzas entre las muestras son diferentes. Comprobemos que la variable **Age** sigue una distribución normal primero aplicando el *F-test* mediante el comando var.test

```
var.test(Mujeres$Age, Hombres$Age)
```

```
##
##  F test to compare two variances
##
## data:  Mujeres$Age and Hombres$Age
## F = 0.94286, num df = 313, denom df = 576, p-value = 0.5615
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.7780865 1.1490592
## sample estimates:
## ratio of variances
##           0.942863
```

El resultado nos indica que las varianzas de edad en los grupos de mujeres y hombres son iguales.

Aplicamos ahora un test para comparar las medias de los 2 grupos, la hipótesis nula será que no hay diferencias significativas entre la media de edades para hombres y mujeres. Podemos usar el *t-Test* indicando que las varianzas de los grupos son iguales (por el resultado anterior).

```
t.test(Mujeres$Age, Hombres$Age, var.equal = TRUE)
```

```
##
##  Two Sample t-test
##
## data:  Mujeres$Age and Hombres$Age
## t = -3.152, df = 889, p-value = 0.001676
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -5.099182 -1.185745
## sample estimates:
## mean of x mean of y
##  27.54140  30.68386
```

El valor de p-values es inferior a 0.05, por lo que rechazamos la hipótesis nula. Es decir, **la media de edad por género tiene una diferencia significativa**.

Establecemos una nueva hipótesis alternativa: que la media de edad de las mujeres es menor que la de los hombres. Por lo tanto, la hipótesis nula corresponde a que la edad de las mujeres es mayor o igual a la de los hombres:

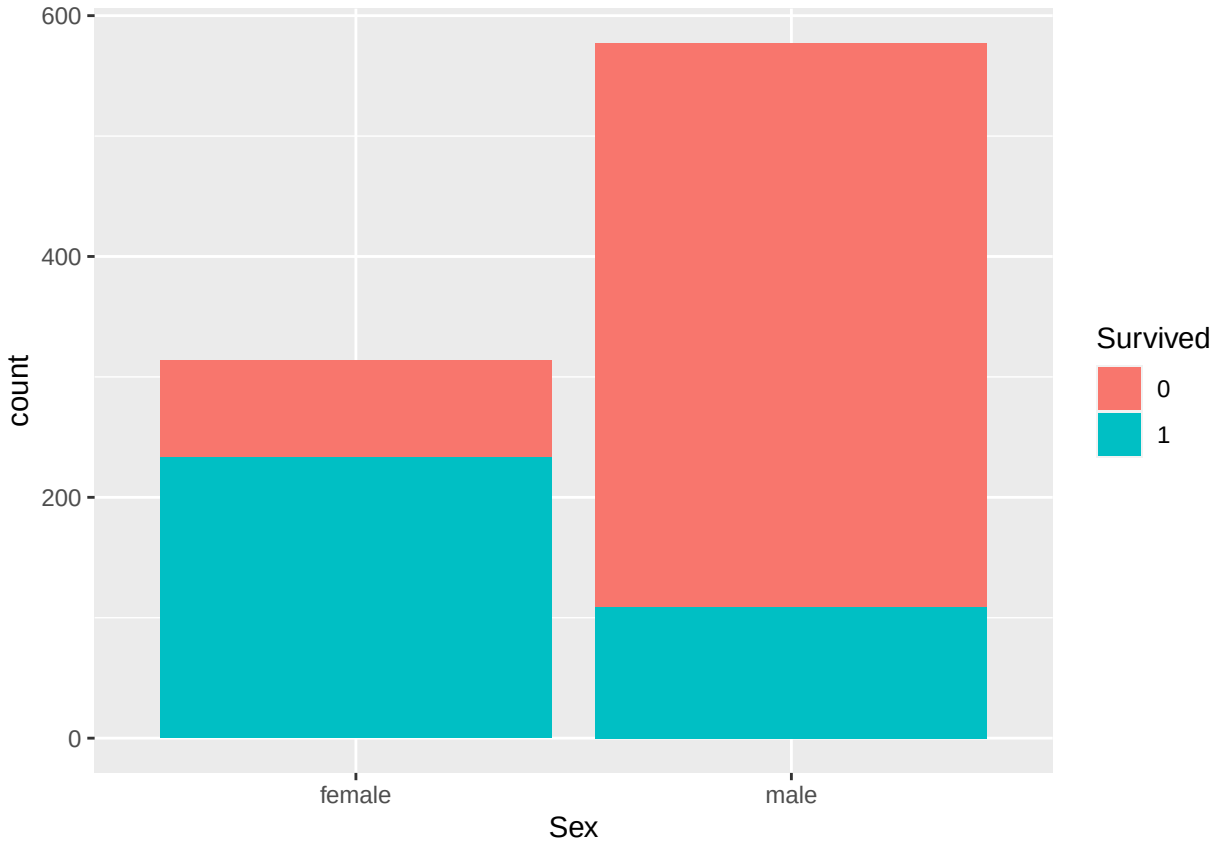
```
t.test(Mujeres$Age, Hombres$Age, alternative = "less", var.equal = TRUE)
```

```
##
##  Two Sample t-test
##
## data:  Mujeres$Age and Hombres$Age
## t = -3.152, df = 889, p-value = 0.000838
## alternative hypothesis: true difference in means is less than 0
## 95 percent confidence interval:
##       -Inf -1.500859
## sample estimates:
## mean of x mean of y
##  27.54140  30.68386
```

Con un p-value menor de 0.5 volvemos a rechazar la hipótesis nula, es decir, podemos afirmar que la media de edad de las mujeres del Titanic es inferior a la media de edad de los hombres a bordo.

4.3.1.2 Survival y Sex

Si hacemos una inspección visual de la relación entre las dos variables:



Parece indicar que ser mujer tenía mas posibilidades de supervivencia que siendo hombre. Vamos a confirmar esto realizando algunos contrastes de hipótesis.

Primero, vamos a comprobar la relación entre la supervivencia y el género, analizando si el género fue un factor importante a la hora de la supervivencia, o dicho de otro modo, si dependiendo de si se era mujer u hombre las posibilidades de supervivencia cambiaban significativamente. Para hacerlo aplicamos el *test exacto de Fisher* que analiza tablas de contingencia. En este caso la hipótesis nula es que la proporción de mujeres que mueren coincide con la proporción de hombre que mueren en el accidente del Titanic.

```
fisher.test(table(data$Sex, data$Survived))
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  table(data$Sex, data$Survived)
## p-value < 2.2e-16
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  0.0575310 0.1138011
## sample estimates:
## odds ratio
## 0.08128333
```

El resultado no deja lugar a dudas con un p-valor < 0.05 rechazamos la hipótesis nula, es decir: **hay diferente proporción de supervivencia entre hombres y mujeres.**

Una vez confirmado que el género si que es importante a ala hora de las probabilidades de supervivencia, vamos a ver quien tiene más probabilidades de supervivencia, si las mujeres o los hombres. Para ello añadimos

una condición a la hipótesis alternativa, usamos "less" para indicar (en la hipótesis nula) que el primer grupo (en este caso las mujeres ya que en orden alfabético es female y male) tiene menor proporción (probabilidad) si se rechaza la hipótesis nula. Es decir, la hipótesis nula dice que "la proporción de mujeres que mueren es mayor que la de los hombres".

```
fisher.test(table(data$Sex, data$Survived), alternative = 'less')
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  table(data$Sex, data$Survived)
## p-value < 2.2e-16
## alternative hypothesis: true odds ratio is less than 1
## 95 percent confidence interval:
##  0.0000000 0.1081391
## sample estimates:
## odds ratio
## 0.08128333
```

De nuevo rechazamos la hipótesis nula (p-value inferior a 0.05) con lo que se cumple la hipótesis alternativa: **la proporción de mujeres que mueren es inferior a la de los hombres**.

Si tenemos en cuenta que la tabla de contingencia que estamos evaluando es la siguiente:

```
table(data$Sex, data$Survived)
```

```
##
##            0   1
##   female  81 233
##   male   468 109
```

El odd ratio de la tabla de contigencia lo calculamos así:

```
tab.contigencia = table(data$Sex, data$Survived)
odd_ratio = (tab.contigencia[1,1] / tab.contigencia[1,2]) /
  (tab.contigencia[2,1] / tab.contigencia[2,2])
odd_ratio
```

```
## [1] 0.08096732
```

Como vimos antes en el test de Fisher, la hipótesis alternativa se hace verdadera cuando el odd_ratio se vuelve 1. Luego cuando a la hipótesis alternativa le ponemos "less", esta se hace verdadera cuando odd_ratio es menor que 1. Y si ponemos "greater" la hipótesis alternativa se hace verdadera si el odd_ratio es mayor que 1.

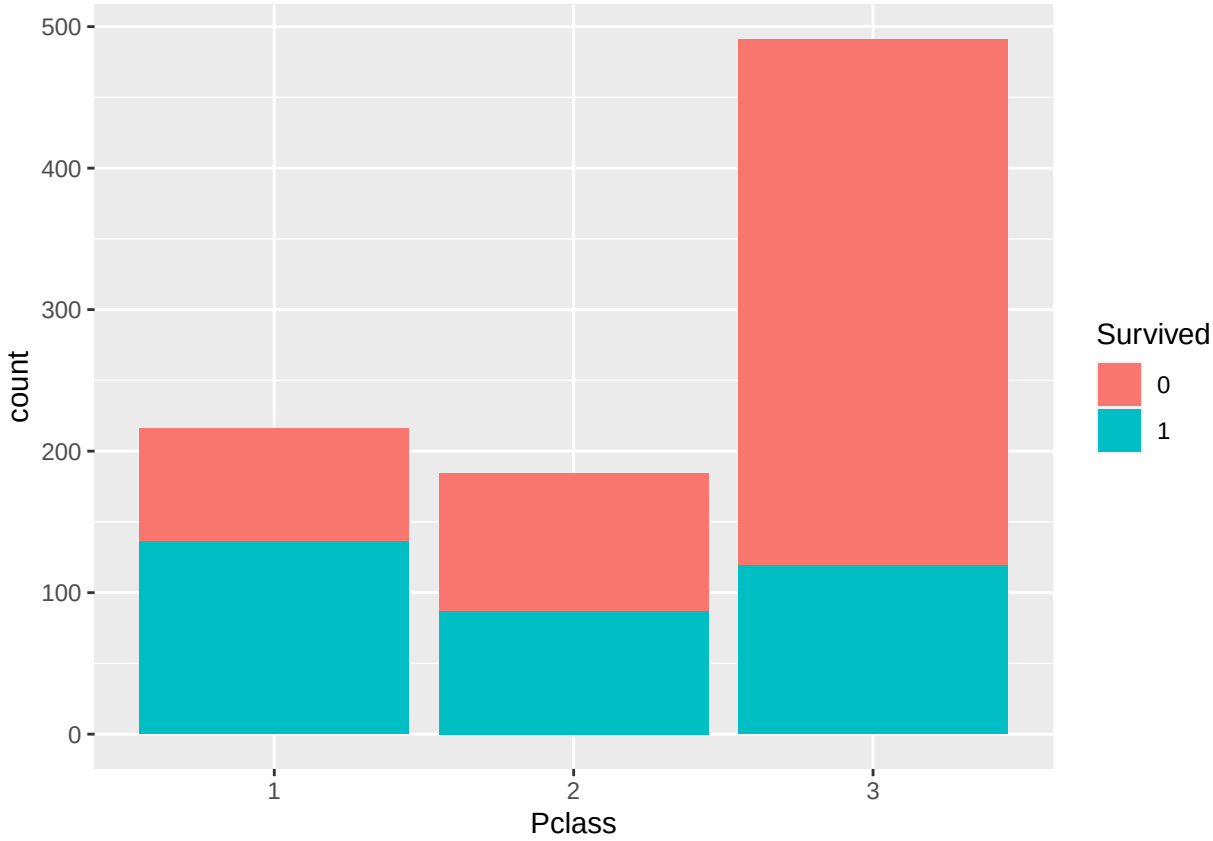
Podemos obtener intervalos de confianza, por ejemplo, al 99% de que la media de las edades de diferentes muestras, van a estar entre los valores 28.8 y 31.26 de media de edad.

```
test = t.test(data$Age, conf.level = 0.99)
test$conf.int
```

```
## [1] 28.34081 30.81203
## attr(,"conf.level")
## [1] 0.99
```

4.3.1.3 Survival y Pclass

De nuevo hacemos una primera inspección visual



En el gráfico se aprecia claramente que el porcentaje de supervivencia por clases disminuye desde la primera hasta la tercera clase.

Para realizar el análisis numéricamente, al ser dos variables categóricas vamos a aplicar por ejemplo el test *Chi-Cuadrado*. Aquí la hipótesis nula nos dice que las variables son independientes. Vamos a comprobarlo:

```
test_chisq <- chisq.test(data$Pclass, data$Survived)
test_chisq
```

```
##
##  Pearson's Chi-squared test
##
## data:  data$Pclass and data$Survived
## X-squared = 102.89, df = 2, p-value < 2.2e-16
```

El p-value obtenido es inferior a 0.05, es decir, rechazamos la hipótesis nula, y por tanto afirmamos que las variables no son independientes. Esto quiere decir que **el grado de supervivencia era distinto dependiendo de la clase en la que estuvieses embarcado**. Confirma el análisis visual, que nos aporta también que las clases con mas superviencia eran las mejores.

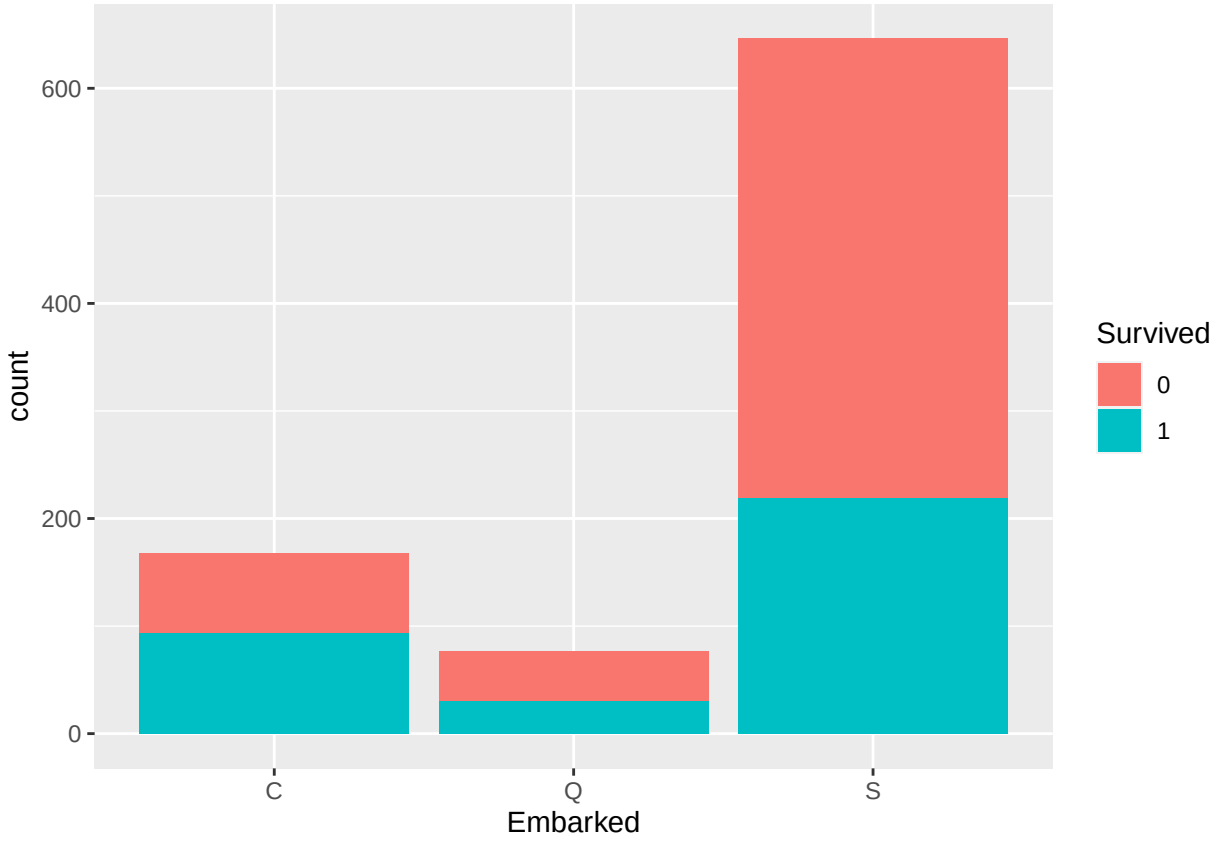
También se puede observar el valor de df que son los grados de libertad, que lo podemos obtener a través de la tabla de contingencias y el estadístico de contraste. Si el estadístico de contraste obtenido supera el valor crítico entonces se rechaza la hipótesis nula.

```
df_chisq <- (length(levels(data$Pclass)) -1 ) * (length(levels(data$Survived)) - 1)
valorcritico_chisq <- qchisq(p=0.05, df = df_chisq, lower.tail = FALSE)
cat(sprintf("El estadístico obtenido es: %f y el valor crítico es: %f", test_chisq$statistic, valorcriti
```

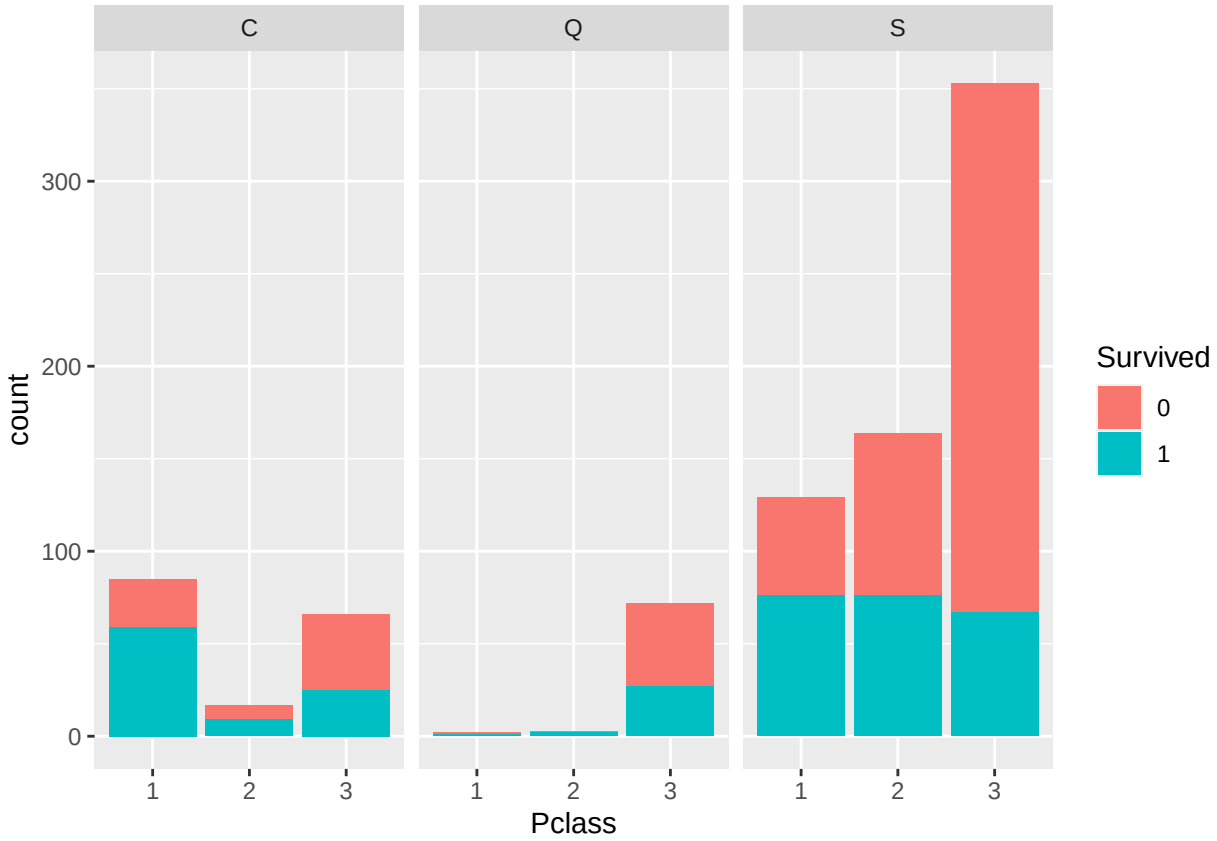
```
## El estadístico obtenido es: 102.888989 y el valor crítico es: 5.991465
```

4.3.1.4 Survived y Embarked

Realizamos el mismo análisis



El gráfico parece indicar que los embarcados en Southampton tenían menos posibilidades de sobrevivir. Puede ser que esté relacionado con la *Pclass* de cada lugar de embarque:



Este nos confirma los embarcados en *S* eran mayoritariamente de clase 3 por lo que tiene sentido que su ratio de supervivencia sea menor.

Vemos ahora el análisis con contrastes de hipótesis.

Primero analizamos si hay relación entre las variables **Embarked** y **Survived** usando de nuevo la *Chi-Squared*

```
test_chisq <- chisq.test(data$Embarked, data$Survived)
test_chisq
```

```
##
##  Pearson's Chi-squared test
##
## data:  data$Embarked and data$Survived
## X-squared = 25.964, df = 2, p-value = 2.301e-06
```

```
df_chisq <- (length(levels(data$Embarked)) -1 ) * (length(levels(data$Survived)) - 1)
valorcritico_chisq <- qchisq(p=0.05, df = df_chisq, lower.tail = FALSE)
cat(sprintf("El estadístico obtenido es: %f y el valor crítico es: %f", test_chisq$statistic, valorcriti
```

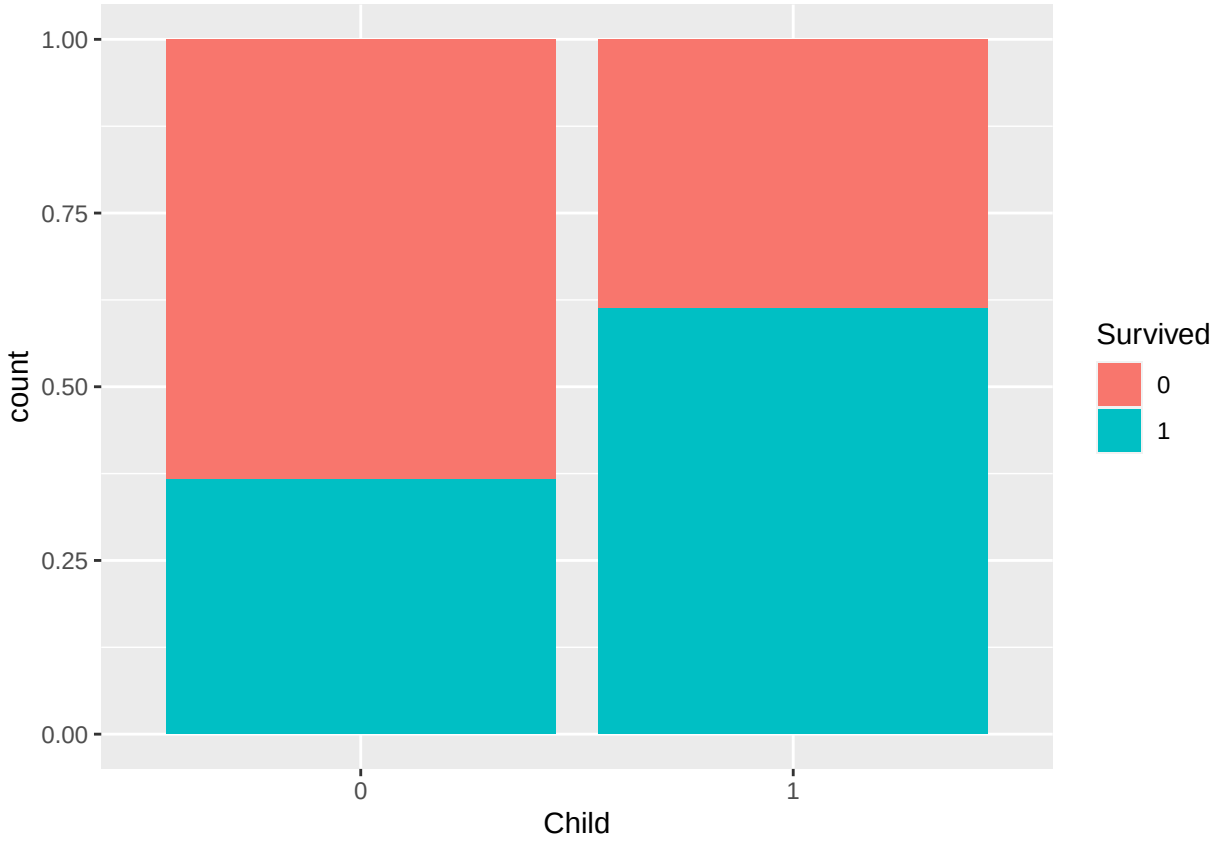
```
## El estadístico obtenido es: 25.964453 y el valor crítico es: 5.991465
```

De nuevo se rechaza la hipótesis nula, por lo tanto **las variables son dependientes**.

4.3.1.5 Survived y Child

Analizamos ahora la nueva variable creada **Child** que corresponde a los niños de edad menor o igual a 8 años (que es un 1 en el valor de la clase)

De nuevo un análisis visual primero:



Parece que si que hay más esperanza de supervivencia en ese caso.

Aplicamos el *test de Fisher* para comprobar si tienen las mismas probabilidades de supervivencia que el resto de pasajeros.

```
fisher.test(table(data$Child, data$Survived))
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  table(data$Child, data$Survived)
## p-value = 0.0002054
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  1.562489 4.857750
## sample estimates:
## odds ratio
##   2.731134
```

Se rechaza la hipótesis nula, es decir que **las proporciones no coinciden**. Analizamos ahora si las probabilidades de supervivencia son mayores o menores. Para ello analizamos si la proporción de no-niños que no sobreviven es menor a la proporción de niños que no sobreviven:

```
fisher.test(table(data$Child, data$Survived), alternative = 'greater')
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  table(data$Child, data$Survived)
```

```
## p-value = 0.0001302
## alternative hypothesis: true odds ratio is greater than 1
## 95 percent confidence interval:
##  1.697776      Inf
## sample estimates:
## odds ratio
##   2.731134
```

Se rechaza la hipótesis nula, y se acepta la alternativa, es decir que la **probabilidad de muerte de los no-niños es mayor que la de los niños**.

```
tab.contigencia = table(data$Child, data$Survived)
tab.contigencia
```

```
##
##       0   1
##   0 525 304
##   1  24  38
```

```
odd_ratio = (tab.contigencia[1,1] / tab.contigencia[1,2]) /
  (tab.contigencia[2,1] / tab.contigencia[2,2])
odd_ratio
```

```
## [1] 2.734375
```

4.3.1.6 Survived y Age

Vamos a analizar ahora si la media de edad de las personas que murieron es coincidente con la media de edad que sobrevivieron. Aplicamos el test *t-Test*

```
t.test(data$Age[which(data$Survived==0)], data$Age[which(data$Survived == 1)])
```

```
##
##  Welch Two Sample t-test
##
## data:  data$Age[which(data$Survived == 0)] and data$Age[which(data$Survived == 1)]
## t = 1.9171, df = 708.56, p-value = 0.05563
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.04569495  3.83775803
## sample estimates:
## mean of x mean of y
##  30.30419  28.40816
```

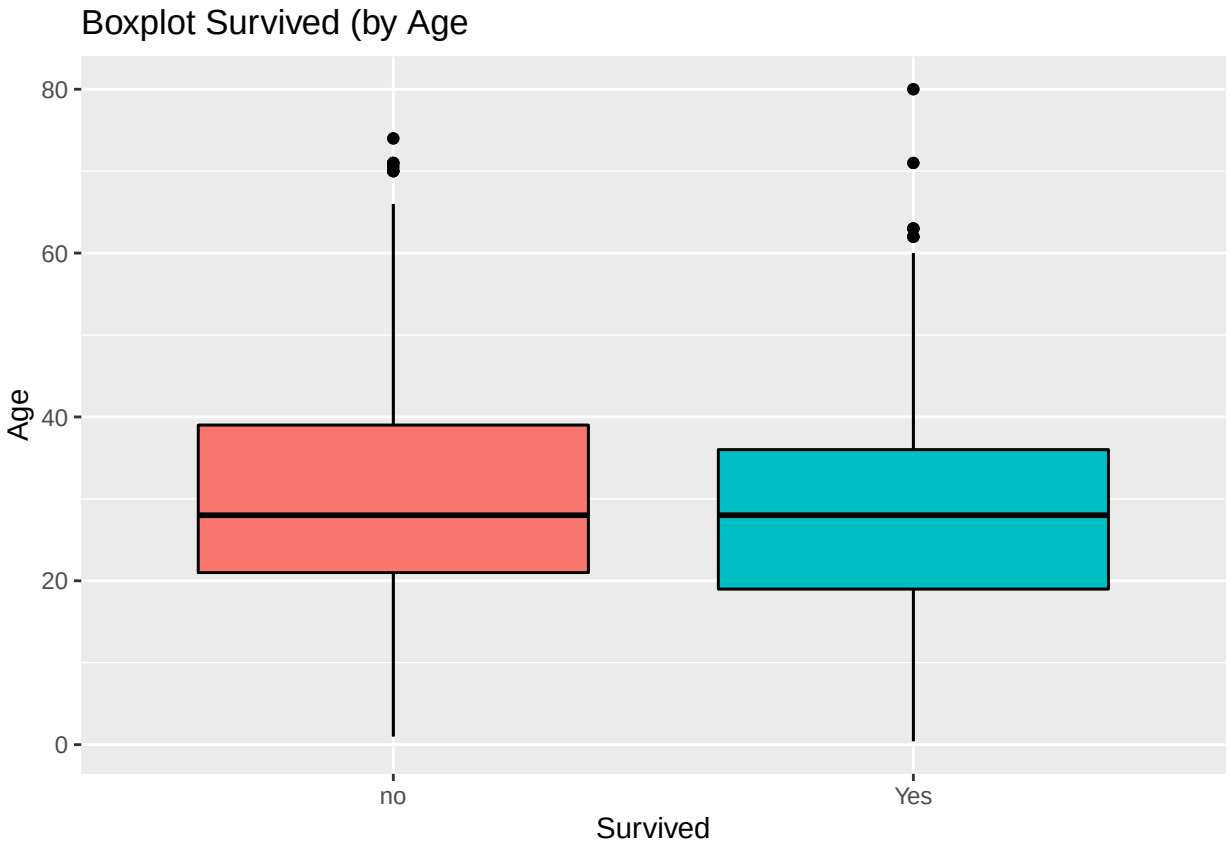
Se observa que el p-valor es inferior a 0.05, con lo que rechazamos la hipótesis nula (que las medias de edad son coincidentes para fallecidos y supervivientes), pero como ya sabíamos, la distribución de **Age** no sigue exactamente una normal, por lo que vamos a aplicar el *test no paramétrico U de Mann-Whitney*

```
wilcox.test(x = data$Age[which(data$Survived==0)],
            y = data$Age[which(data$Survived==1)],
            paired =FALSE
            )
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  data$Age[which(data$Survived == 0)] and data$Age[which(data$Survived == 1)]
## W = 98342, p-value = 0.2321
## alternative hypothesis: true location shift is not equal to 0
```

La prueba no-paramétrica nos devuelve un p-value por encima de 0.05 y por tanto no rechazamos la hipótesis nula, con lo que podemos decir que **la media de edad es coincidente entre los muertos y los supervivientes del accidente del Titanic**.

Gráficamente podemos comprobarlo mediante un Boxplot



Se ve claramente que las distribuciones son muy parecidas.

4.3.1.7 Age y Embarked Para analizar si **Age** es independiente del puerto de embarque **Embarked** , podemos realizar un test Anova, donde la hipótesis nula nos dice que la media de la edad es independiente del puerto de embarque, es decir que la media de edad de la gente que subió en "C" (Cherbourg), coincide con la media de los que embarcaron en "Q" (Queenstown) y también con los que embarcaron en "S" (Southampton). Y por el contrario la hipótesis nula es que alguna de las medias es diferente.

```
modelo_anova <- aov(formula = Age ~ Embarked, data =data )
resumen_anova <- summary(modelo_anova)
resumen_anova
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## Embarked      2    605   302.3   1.482  0.228
## Residuals   888 181086   203.9
```

Observamos dos filas, una que pone "Embarked" y otra que pone "Residuals". La primera fila corresponde a todo lo relativo a la varianza explicada (la de la variable independiente **Embarked**) y la segunda fila relativo a la varianza no explicada o residual.

Explicación de los resultados del test Anova:

Df son los grados de libertad, para el caso de la varianza explicada (la de la variable independiente) es k-1.

En nuestro caso como “Embarked” tiene 3 valores posibles, entonces Df=2.

Para la parte de la varianza residual, es n - k, por tanto el valor Df=888 (891 son los registros del dataframe, si le restamos los 2 de antes quedan los 888).

Sum Sq es la suma de la diferencia de los cuadrados, conocidos como SCDe (variación entre-grupos) y SCDi (variación intra-grupos). La función nos ha devuelto 427 y 175873 respectivamente.

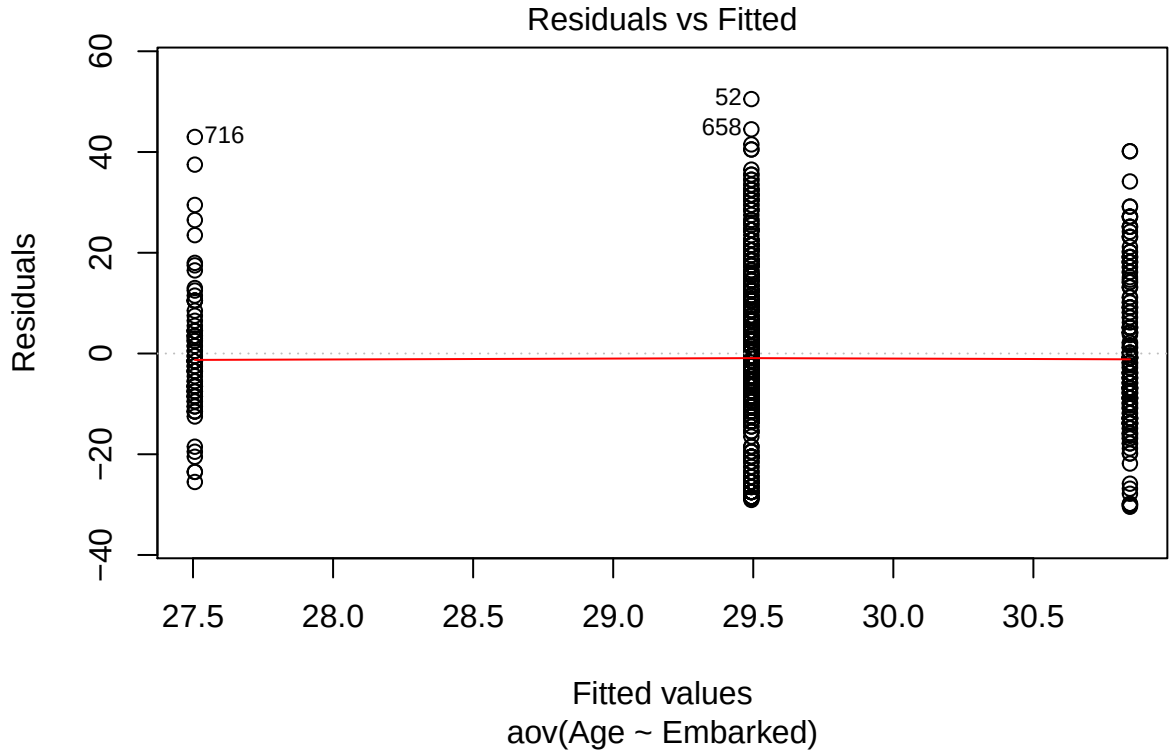
Mean Sq es la media cuadrática referente a entre-grupos e intra-grupos. Lo que también llamamos como Varianza explicada (línea superior, es decir 213.6) y varianza no explicada (línea inferior, es decir 198.1). Corresponde al cociente entre la suma de diferencias de cuadrados y los grados de libertad, es decir realmente Mean Sq = Sum Sq / Df lo que se puede comprobar realizando los cálculos sobre los datos anteriores.

F Es el estadístico (Fisher-Snedecor), que es el cociente entre la varianza explicada entre la varianza no explicada. Por tanto es el cociente entre los valores Mean Sq, siendo el numerador la fila superior (213.6) y el denominador la fila inferior (213.6). El valor del cociente es 0.956, es decir el estadístico F.

Pr(>F) Es la probabilidad asociada, el p-valor que nos indicará si se rechaza o no la hipotesis nula. En este caso el valor es: 0.442 que es mayor que 0.05 y nos permite aceptar (no rechazar) la hipótesis nula que decía que las medias de la variable “Age” en función del valor de “Embarked” eran iguales.

Revisamos los distintos gráficos que nos proporciona anova

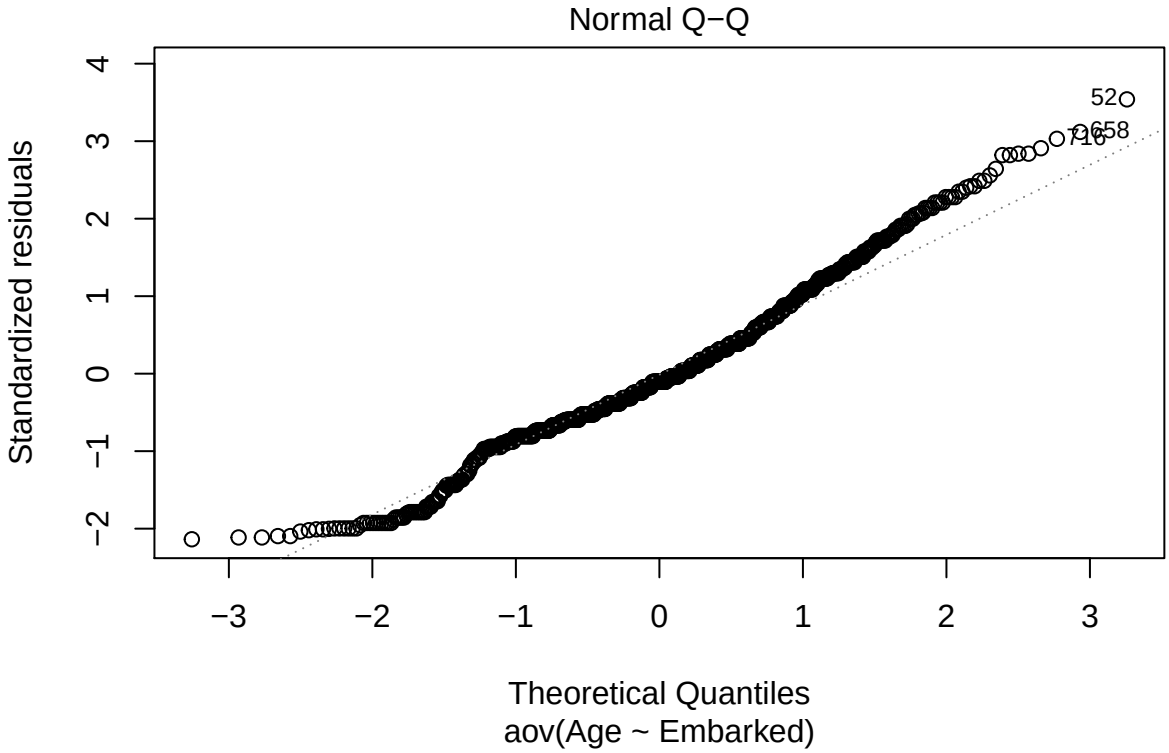
Gráfico Residuals vs Fitted



Esta gráfica nos muestra si los residuos tienen patrones no lineales. Puede existir una relación no lineal entre las variables explicativas y la variable explicada, y ese patrón podría verse en esta gráfica si el modelo no captura esa relación no lineal. Si los residuos están igualmente distribuidos alrededor de una línea horizontal sin patrones diferentes, es una buena señal de que no hay relaciones no lineales.

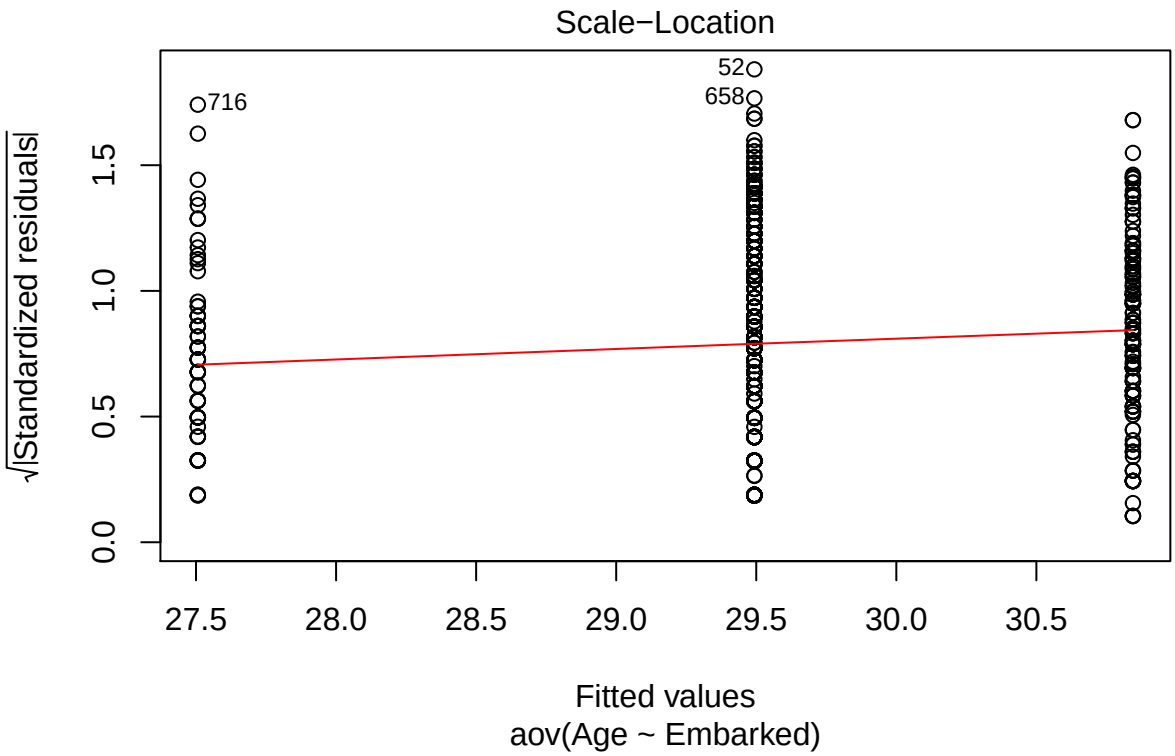
En nuestro caso concreto, se ve una línea casi horizontal (hay una mini curva) y los residuos se distribuyen alrededor de dicha línea. Podemos decir que no hay indicios de relaciones no lineales.

Gráfico Normal Q-Q



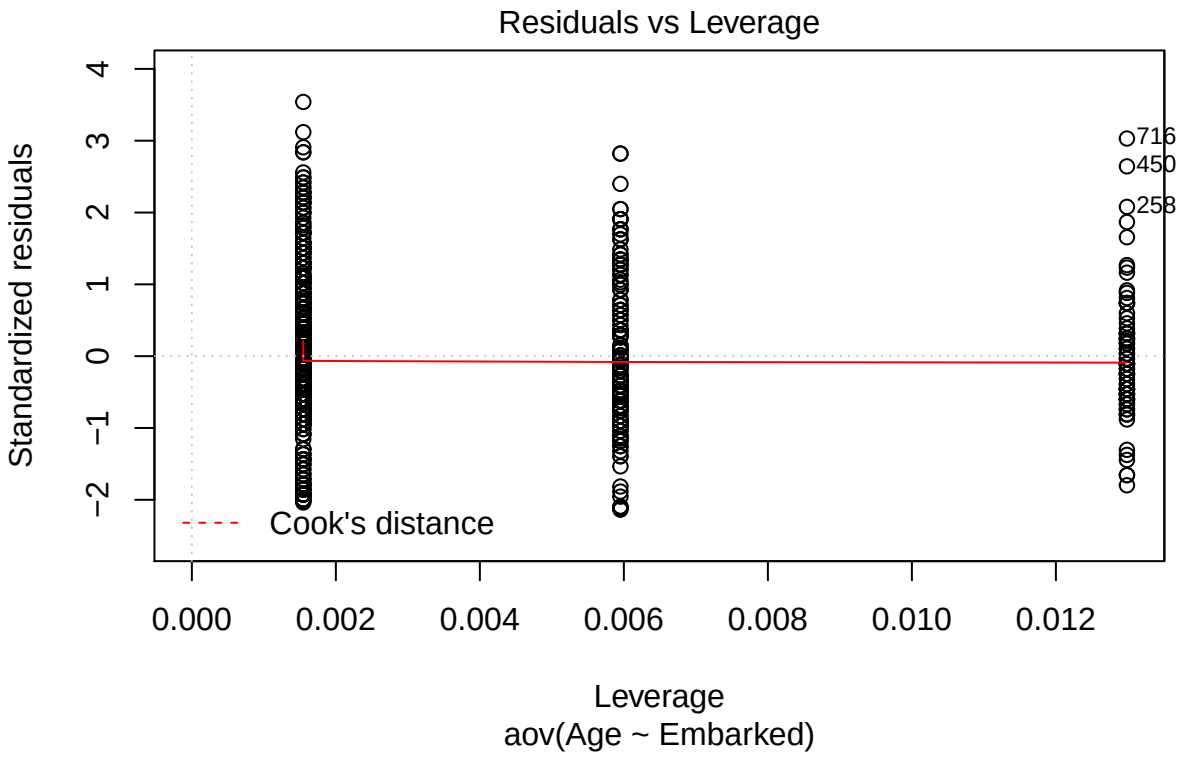
Si no nos ponemos muy estrictos en la exigencia, con el gráfico *Normal Q-Q* podemos ver que la variable dependiente sigue una distribución aproximadamente normal. La mayoría de los puntos se encuentran alineados en la línea de puntos y solamente es en los extremos donde esa situación no se cumple.

Gráfica Scale-Location



La gráfica *Scale-Location* también es conocida con el nombre de gráfica *Spread-Location*. Aquí vemos si los residuos se reparten de forma equitativa.En nuestro ejemplo, vemos casi una recta horizontal y los puntos se distribuyen de forma muy similar a los lados de la línea y en los extremos también parece un gráfico simétrico (tanto en horizontal como en vertical). Es decir que el gráfico nos está indicando que se cumple la homocedasticidad en nuestro modelo.

Gráfica Residuals vs Leverage

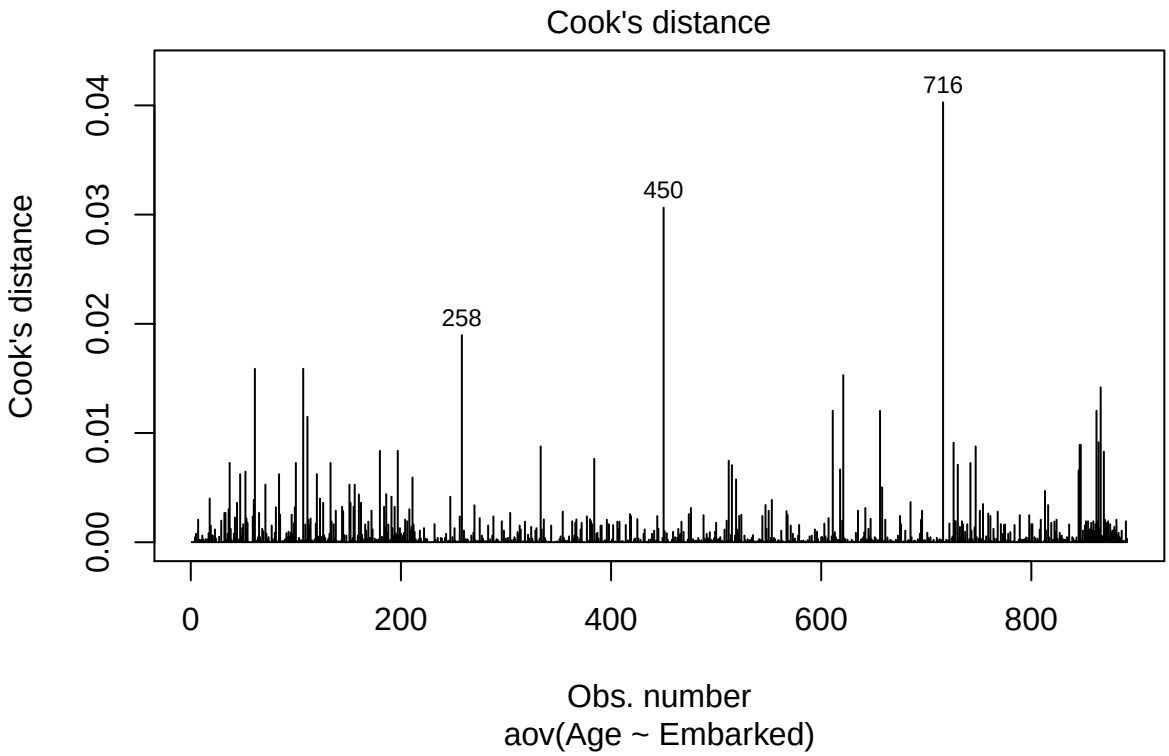


Esta gráfica sirve para ayudarnos a encontrar casos influyentes (es decir, sujetos) si es que los hay. No todos los valores atípicos son influyentes en el análisis de regresión lineal. En el caso de existir valores extremos (outliers), puede que no sean influyentes para determinar la línea de regresión. Eso significa que los resultados no serían muy diferentes si los incluimos o excluimos del análisis, es decir no son influyentes. Por otra parte, algunos casos podrían ser muy influyentes, incluso aunque parezca que están dentro de un rango razonable de los valores. Pueden ser casos extremos contra una línea de regresión y pueden alterar los resultados si los excluimos del análisis. Es decir, estos no están alineados (tendencia) con la mayoría de los casos.

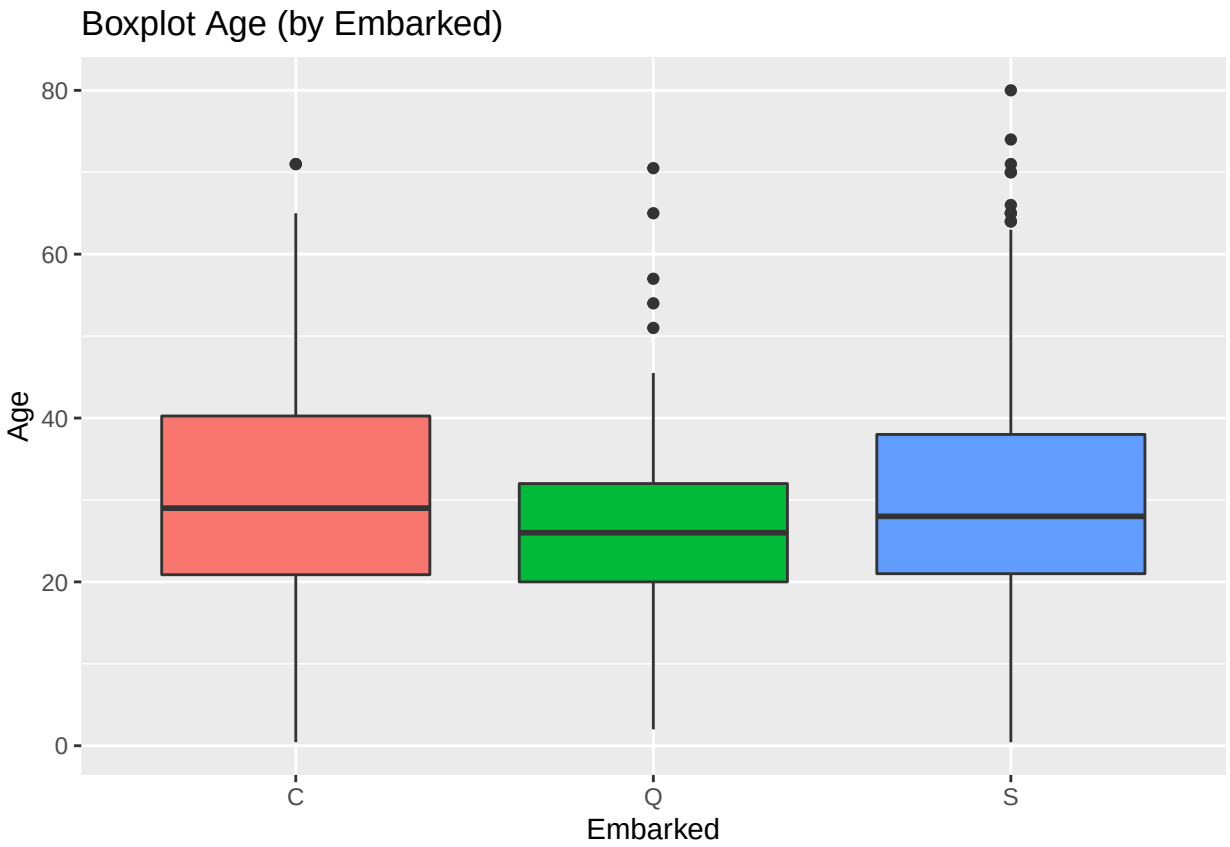
Tenemos que buscar casos que estén fuera de una línea discontinua (que es la distancia de Cook). Cuando encontremos casos fuera de la distancia de Cook (es decir que tienen puntuación alta de distancia de Cook) se consideran casos influyentes en los resultados de la regresión. Y esa regresión se verá afectada si excluimos esos casos.

Grafica Distancia de Cook

Otra forma de mostrar la distancia de Cook es usando el gráfico que proporciona Anova



Después de este análisis y dando por aceptadas las condiciones del cumplimiento de Anova, podemos afirmar que **las medias de edad en los diferentes puertos de Embarke es coincidente**. Y de hecho, si vemos el Boxplot de la variable **Age** en función de la variable **Embarked**:



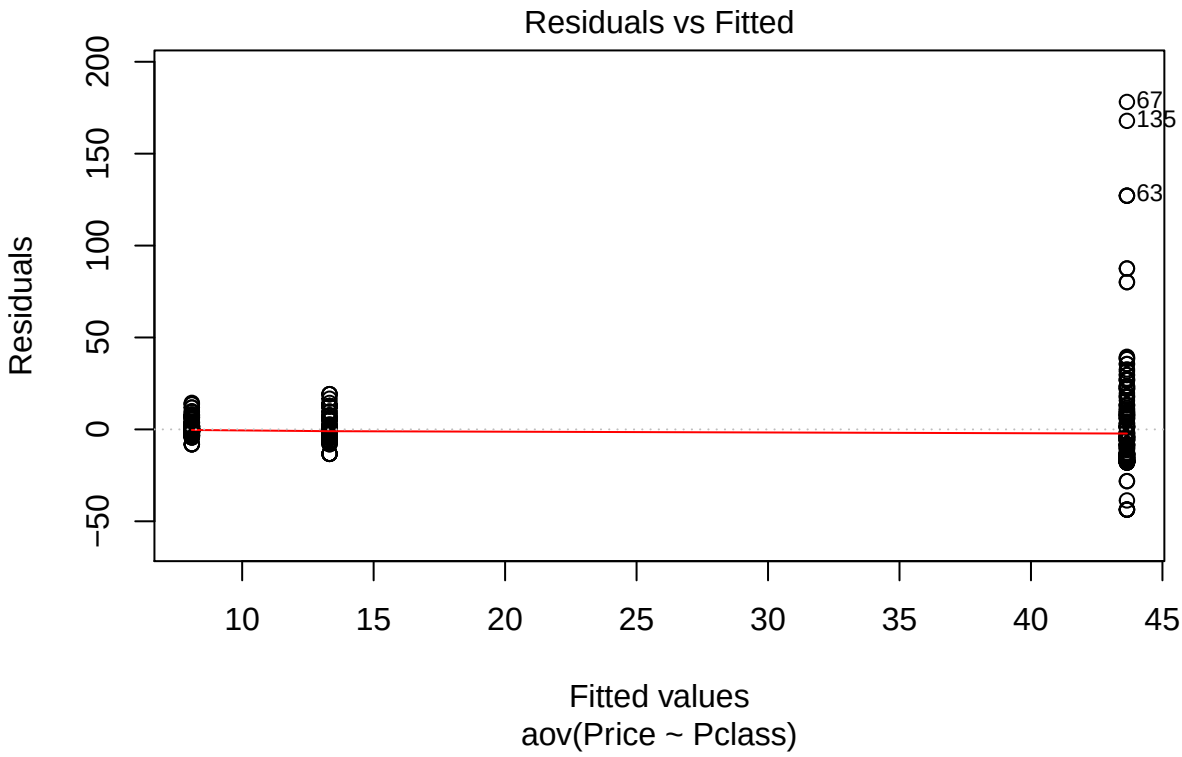
4.3.1.8 Price y Pclass

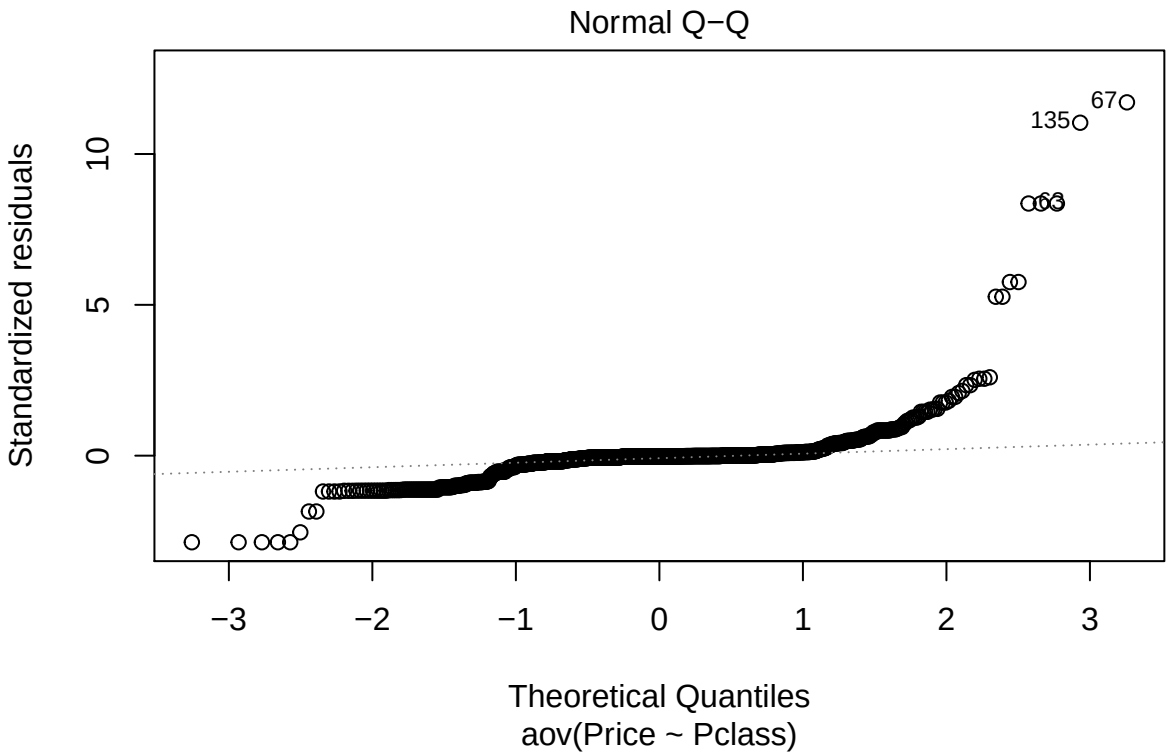
Intentamos realizar Anova con estas dos variables, usando como hipótesis nula que el precio medio pagado es igual para cada una de las 3 clases

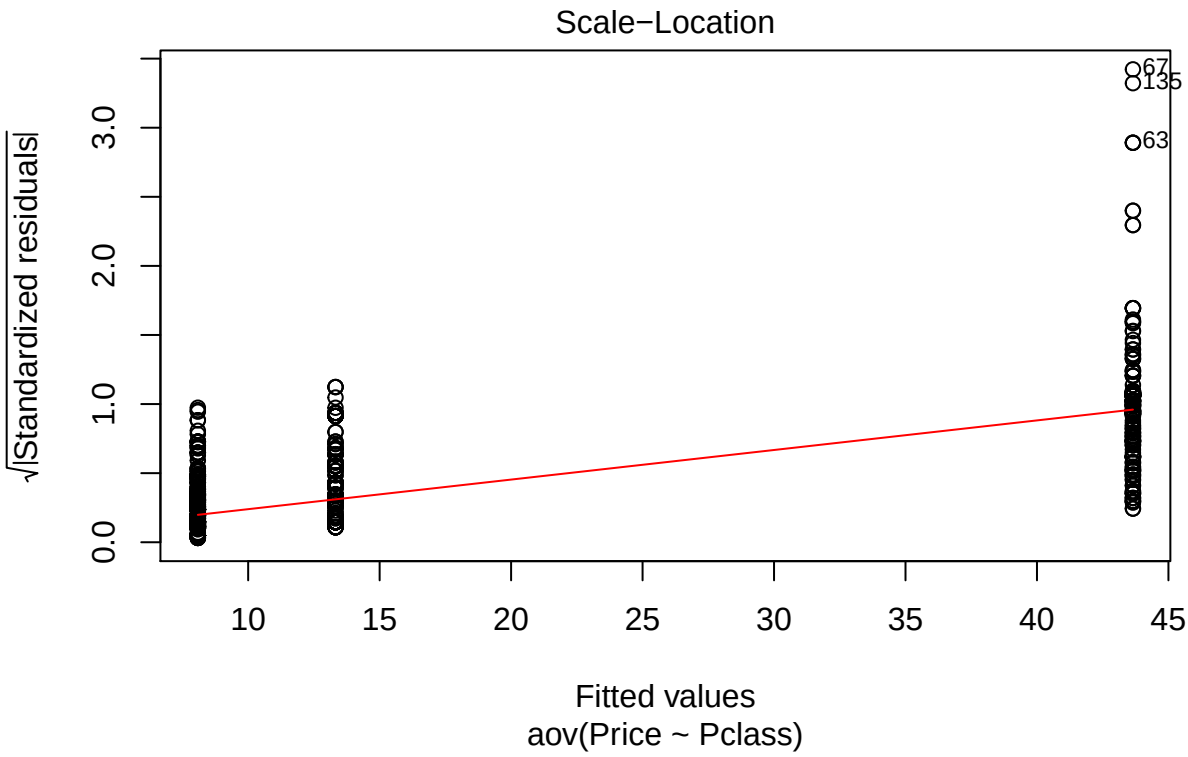
```
modelo_anova <- aov(formula = Price ~ Pclass, data =data )
resumen_anova <- summary(modelo_anova)
resumen_anova
```

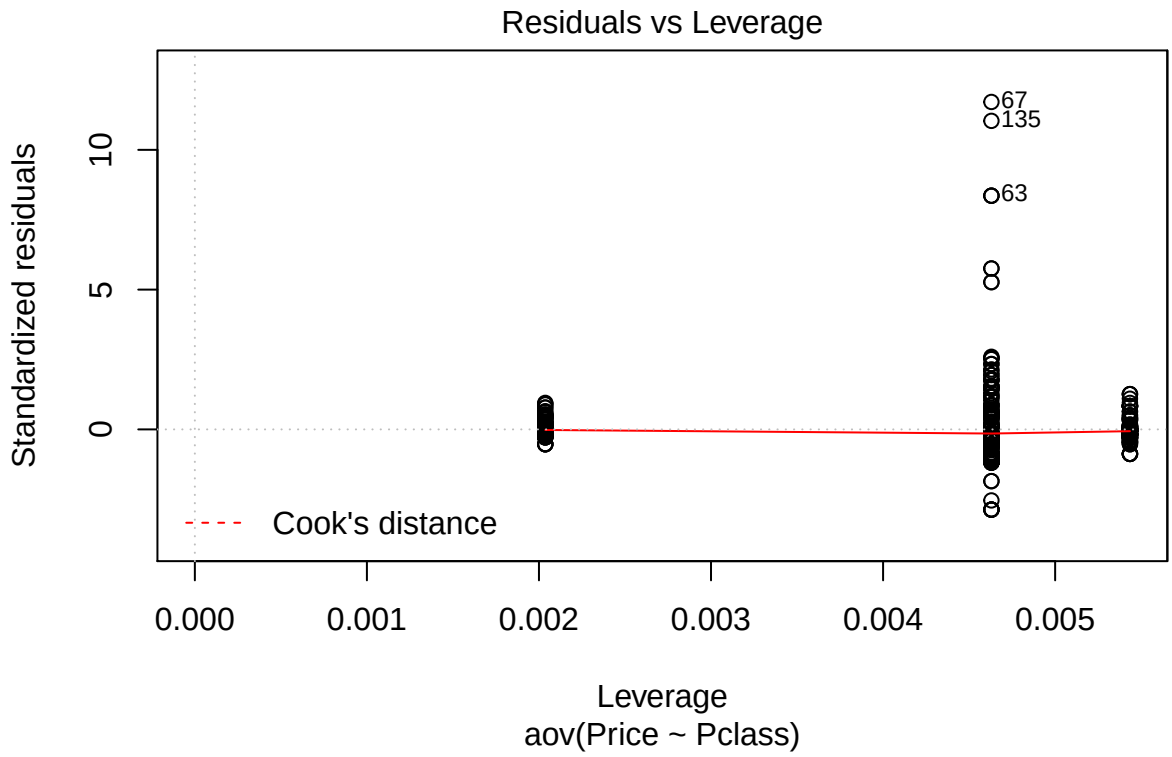
```
##              Df Sum Sq Mean Sq F value Pr(>F)
## Pclass        2 194362   97181   418.3 <2e-16 ***
## Residuals   888 206326     232
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Si mostramos los gráficos:


Residuals vs Fitted
Residuals
200
150
100
50
0
−50
67
135
63
10
15
20
25
30
35
40
45
Fitted values
aov(Price ~ Pclass)


Normal Q−Q
Standardized residuals
0
5
10
−3
−2
−1
0
1
2
3
67
135
Theoretical Quantiles
aov(Price ~ Pclass)


Scale−Location
√|Standardized residuals|
0.0
1.0
2.0
3.0
10
15
20
25
30
35
40
45
67
135
63
Fitted values
aov(Price ~ Pclass)



Observamos que no se cumplen las condiciones para aplicar Anova y el test no tiene valor.

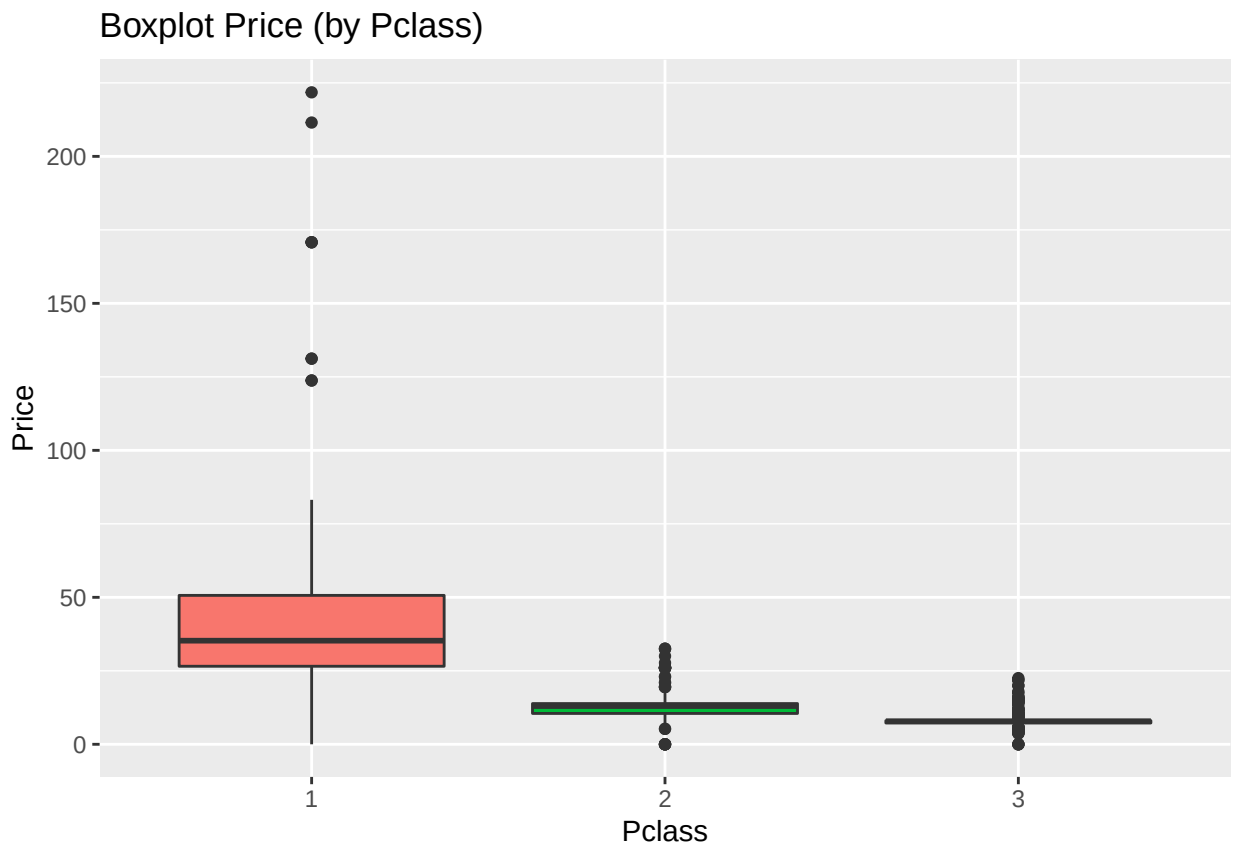
Vamos a intentar usar un test no paramétrico, *Kruskal Wallis*

```
modelo_kruskal <- kruskal.test(formula = Price ~ Pclass, data =data )
modelo_kruskal
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  Price by Pclass
## Kruskal-Wallis chi-squared = 566.65, df = 2, p-value < 2.2e-16
```

El resultado de este test arroja un p-value inferior a 0.05 y por tanto rechazamos la hipótesis nula. Es decir **el precio medio pagado NO es igual para cada una de las 3 clases** lo cual era el resultado esperado.

Lo podemos comprobar también visualmente con el boxplot relativo a ello:



4.3.2 Modelos

Vamos a aplicar diversos modelos predictivos de la variable **Survived**.

Vamos a utilizar como variables explicativas las variables que traía por defecto el conjunto de datos original (menos las que hemos eliminado en el proceso de transformación y limpieza de datos ya comentado).

Creamos un conjunto de entrenamiento y un conjunto de test. Es decir,vamos a realizar una partición, un 75% de datos para el entrenamiento, y el 25% restante para validar los modelos.

```
set.seed(666)

split = sample.split(data$Survived, SplitRatio = 0.75)
data_train = subset(data, split == TRUE)
data_test = subset(data, split == FALSE)
```

Creamos una función que nos imprima la matriz de confusión

```
# Función necesaria para imprimir la matriz de confusión a partir de un CrossTable
plot_matriz_confusion <- function(par_crosstable) {
  #Obtenemos la matriz de confusion en porcentaje
  datplotconfusion <- as.data.frame(par_crosstable[2]$prop.row)
  colnames(datplotconfusion) <- c('Observacion','Prediccion','Valor')
  datplotconfusion$Valor <- datplotconfusion$Valor*100
  #Ordenamos los factores de las clases
  datplotconfusion <- datplotconfusion %>% arrange(Observacion)
  clasesord <- sort(as.character(levels(datplotconfusion$Observacion)),decreasing = FALSE)
  datplotconfusion$Observacion <- factor(datplotconfusion$Observacion, levels = clasesord)
```

```
  #Ordenar las clases descendente (para el plot)
  clasesinv<-rev(as.character(unique(datplotconfusion$Observacion)))

  plot1 <- ggplot() +
    geom_tile(aes(x=Observacion, y=Prediccion,fill=Valor),
              data=datplotconfusion, color="black",size=0.1) +
    labs(x="Observación real (%)",y="Predicción (%)") +
    scale_y_discrete(limits=clasesinv)

  plot1 = plot1 +
    geom_text(aes(x=Observacion, y=Prediccion, label=sprintf("%.2f", Valor)),
              data=datplotconfusion, size=3, colour="black") +
    scale_fill_gradient(low="gray",high="red")

  #Obtenemos la matriz de confusion en numero de muestras
  datplotconfusion <- as.data.frame(par_crosstable[1]$t)
  colnames(datplotconfusion) <- c('Observacion','Prediccion','Valor')
  #Ordenamos los factores de las clases
  datplotconfusion <- datplotconfusion %>% arrange(Observacion)
  clasesord <- sort(as.character(levels(datplotconfusion$Observacion)),decreasing = FALSE)
  datplotconfusion$Observacion <- factor(datplotconfusion$Observacion, levels = clasesord)
  #Ordenar las clases descendente (para el plot)
  clasesinv<-rev(as.character(unique(datplotconfusion$Observacion)))
  plot2 <- ggplot() +
    geom_tile(aes(x=Observacion, y=Prediccion,fill=Valor),
              data=datplotconfusion, color="black",size=0.1) +
    labs(x="Observación real (nº)",y="Predicción (nº)") +
    scale_y_discrete(limits=clasesinv)

  plot2 = plot2 +
    geom_text(aes(x=Observacion, y=Prediccion, label=sprintf("%d", Valor)),
              data=datplotconfusion, size=3, colour="black") +
    scale_fill_gradient(low="gray",high="red")
  return(grid.arrange(plot1, plot2, ncol=2))
}
```

Creamos un dataframe que ira guardando los distintos valores de precisión y errores de clasificación

```
tabla.modelos <- data.frame('Modelo' = character(),
                            'Precisión' = numeric(),
                            'Errores' = numeric(),
                            stringsAsFactors = FALSE)

str(tabla.modelos)
```

```
## 'data.frame':    0 obs. of  3 variables:
##  $ Modelo   : chr
##  $ Precisión: num
##  $ Errores  : num
```

4.3.2.1 Modelo de regresión logística Un modelo de regresión logística es un tipo de análisis de regresión que se utiliza para predecir el resultado de una variable categórica, en función de las variables independientes. En nuestro caso, la variable objetivo **Survived** es una variable categórica binaria, es decir que tomar valor de 0 o 1 (verdadero o falso). La variable toma el valor 1 cuando el pasajero ha sobrevivido

al accidente, y 0 en caso contrario.

4.3.2.1.1 Primer modelo glm

Creamos un primer modelo glm1 con las variables "original": **Pclass**, **SibSp**, **Parch**, **Sex**, **Age**, **Fare** y **Embarked**

```
modelo_glm1 <- glm(formula=Survived~ Pclass+SibSp+Parch+Sex+Age+Fare+Embarked, data = data_train, family
summary(modelo_glm1)
```

```
##
## Call:
## glm(formula = Survived ~ Pclass + SibSp + Parch + Sex + Age +
##     Fare + Embarked, family = binomial(link = "logit"), data = data_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6613  -0.5992  -0.3521   0.6011   2.5974
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.408963   0.576962   7.642 2.14e-14 ***
## Pclass2     -1.059885   0.358265  -2.958  0.00309 **
## Pclass3     -2.372594   0.373921  -6.345 2.22e-10 ***
## SibSp       -0.232708   0.112753  -2.064  0.03903 *
## Parch       -0.112107   0.141833  -0.790  0.42928
## Sexmale     -2.825147   0.238682 -11.836  < 2e-16 ***
## Age         -0.036694   0.008861  -4.141 3.46e-05 ***
## Fare         0.002420   0.003315   0.730  0.46534
## EmbarkedQ   -0.191160   0.458762  -0.417  0.67691
## EmbarkedS   -0.733643   0.290092  -2.529  0.01144 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 889.27  on 667  degrees of freedom
## Residual deviance: 558.04  on 658  degrees of freedom
## AIC: 578.04
##
## Number of Fisher Scoring iterations: 5
```

Se puede observar en este modelo que tanto las variables **Fare** como **Parch** no son significativas, es decir, no están aportando nada a la hora de predecir la variable **Survived**.

Además, podemos comprobar el "sentido" de la significación. Por ejemplo, vemos como significativa la variable dummy. Con las variables categóricas se crean automáticamente tantas variables dummy como niveles – 1. En el caso de la variable Sex, al tener 2 valores ha creado la variable **Sexmale**, es decir la parte correspondiente a hombres, mientras que la parte de mujeres forma parte del Intercept del modelo. Concretamente vemos que **Sexmale** es una variable significativa, pero con valor negativo (de las más negativas junto con **Pclass3**). Eso significa que esas variables son una influencia "negativa" de cara a la supervivencia. Es decir, esas variables contribuyen negativamente a la supervivencia, se puede ver por ejemplo que **Pclass3** afecta más negativamente que **Pclass2**.

Ahora usamos el conjunto de datos de test para validar nuestro modelo. (Para este primer modelo mostramos el código como ejemplo)

```
predict_glm1 <- predict(object=modelo_glm1, newdata=data_test, type = "response")
r1 <- pROC::roc(response=data_test$Survived, predictor = predict_glm1)
```

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

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

```
threshold_r1 <- pROC::coords(r1, "best", ret = "threshold", transpose="FALSE")
predict_glm1 <- if_else(predict_glm1 < threshold_r1[1,1], 0, 1)
mat.confusion_glm1 <- table(data_test$Survived, predict_glm1)
mat.confusion_glm1
```

```
##    predict_glm1
##       0   1
##   0 127  10
##   1  36  50
```

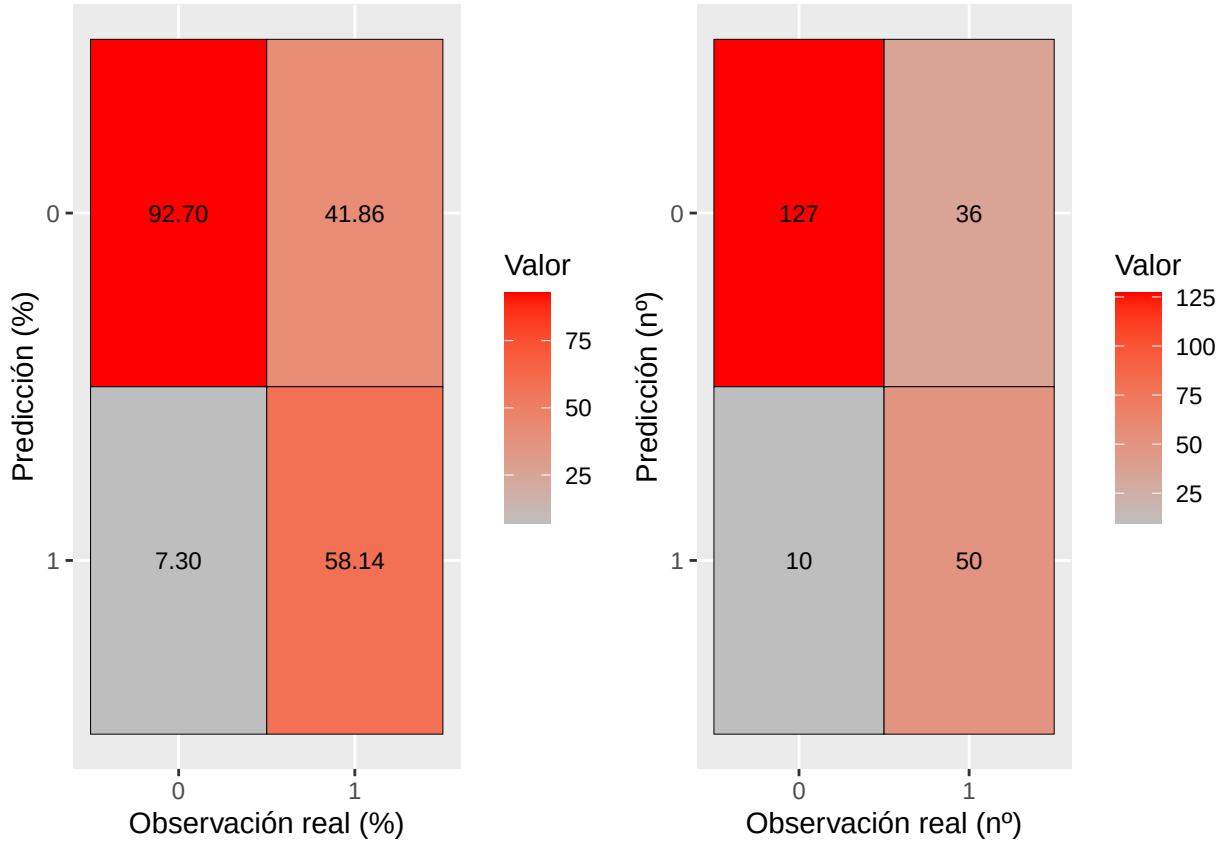
```
pct.correcto_glm1 <- 100 * sum(diag(mat.confusion_glm1)) / sum(mat.confusion_glm1)
print(sprintf("El %% de registros correctamente clasificados es: %.4f %%",
              pct.correcto_glm1))
```

```
## [1] "El % de registros correctamente clasificados es: 79.3722 %"
```

```
csstab_glm1 <- CrossTable(data_test$Survived, predict_glm1,
                          prop.chisq  = FALSE, prop.c = FALSE, prop.r =FALSE,
                          dnn = c('Reality', 'Prediction'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  223
##
##
##              | Prediction
##      Reality |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |       127 |        10 |       137 |
##              |     0.570 |     0.045 |           |
## -------------|-----------|-----------|-----------|
##            1 |        36 |        50 |        86 |
##              |     0.161 |     0.224 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       163 |        60 |       223 |
## -------------|-----------|-----------|-----------|
##
##
```

```
plot_matriz_confusion(csstab_glm1)
```




```
tabla.modelos[nrow(tabla.modelos)+1,] = c("glm1", pct.correcto_glm1, sum(mat.confusion_glm1)-sum(diag(ma
```

4.3.2.1.2 Segundo modelo glm Construimos otro modelo (glm2), similar al anterior pero esta vez partiendo de los resultados del modelo anterior, le vamos a quitar aquellas variables que vimos que no eran significativas para el modelo de predicción. Por lo tanto, tendremos un modelo2 cuyas variables a utilizar serán **Pclass**, **SibSp**, **Sex** y **Age**.

```
modelo_glm2 <- glm(formula=Survived~ Pclass+SibSp+Sex+Age, data = data_train, family = binomial(link =
summary(modelo_glm2)
```

```
##
## Call:
## glm(formula = Survived ~ Pclass + SibSp + Sex + Age, family = binomial(link = "logit"),
##     data = data_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7645  -0.5872  -0.3731   0.5970   2.5376
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.151296   0.469567   8.841  < 2e-16 ***
## Pclass2     -1.407761   0.312280  -4.508 6.54e-06 ***
## Pclass3     -2.606709   0.303732  -8.582  < 2e-16 ***
## SibSp       -0.276434   0.105515  -2.620   0.0088 **
## Sexmale     -2.831046   0.228425 -12.394  < 2e-16 ***
```

```
## Age         -0.037849   0.008797  -4.303 1.69e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 889.27  on 667  degrees of freedom
## Residual deviance: 567.58  on 662  degrees of freedom
## AIC: 579.58
##
## Number of Fisher Scoring iterations: 5
```

Se comprueba que todas las variables utilizadas son variables significativas para el modelo. Además, el modelo parece haber mejorado un poco en cuanto a la predicción:

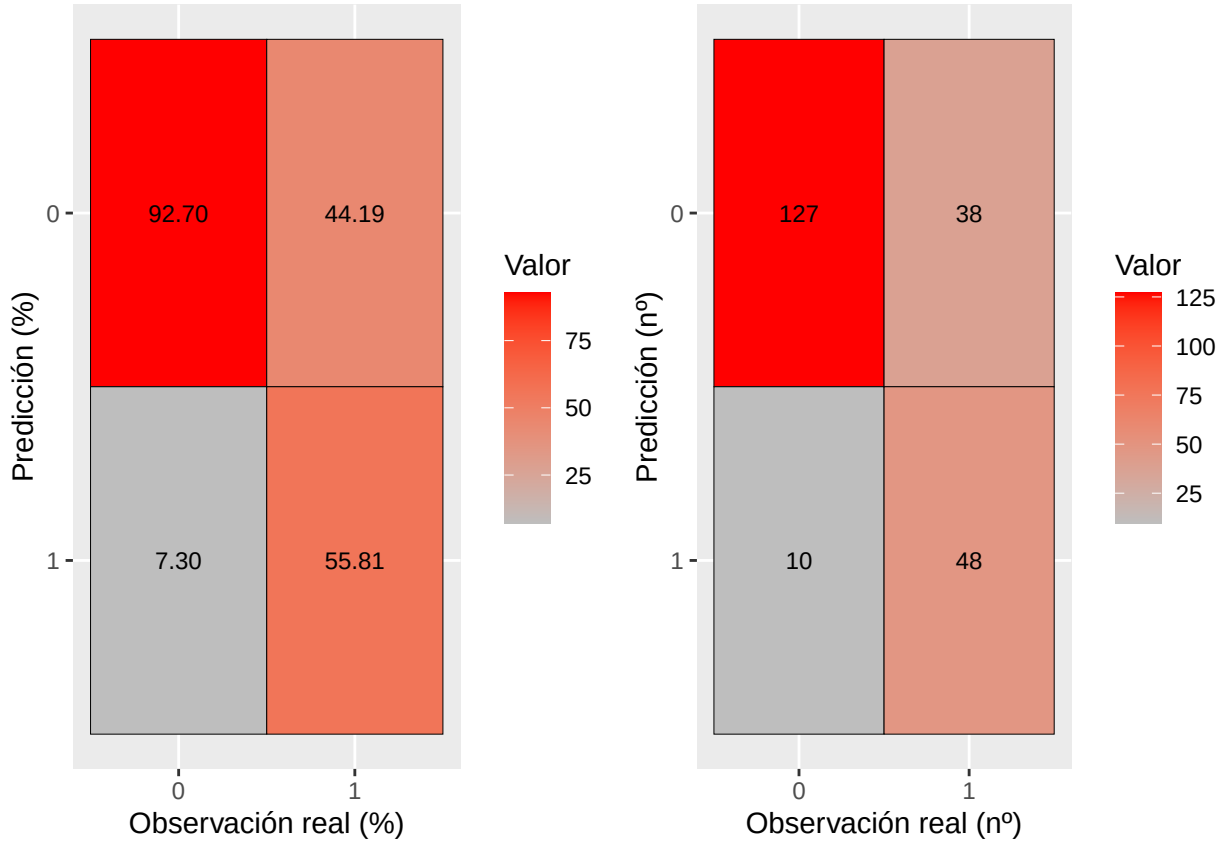
```
## Setting levels: control = 0, case = 1
```

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

```
##    predict_glm2
##       0   1
##   0 127  10
##   1  38  48
```

```
## [1] "El % de registros correctamente clasificados es: 78.4753 %"
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  223
##
##
##              | Prediction
##      Reality |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |       127 |        10 |       137 |
##              |     0.570 |     0.045 |           |
## -------------|-----------|-----------|-----------|
##            1 |        38 |        48 |        86 |
##              |     0.170 |     0.215 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       165 |        58 |       223 |
## -------------|-----------|-----------|-----------|
##
##
```



4.3.2.1.3 Tercer modelo glm

Hasta ahora se han utilizado las variables del conjunto de datos "original". Vamos a construir otro modelo (glm3), pero esta vez vamos a intentar utilizar alguna de las variables que hemos construido para ayudar a predecir la supervivencia. Para este caso vamos a utilizar las variables nuevas **Child** y también **FamilySize**. Además de esas dos variables, vamos a utilizar otras tres variables originales: **Pclass**, **Sex** y **Age**.

```
modelo_glm3 <- glm(formula=Survived~ Sex+Pclass+Age+Child+FamilySize,, data = data_train, family = bino
summary(modelo_glm3)
```

```
##
## Call:
## glm(formula = Survived ~ Sex + Pclass + Age + Child + FamilySize,
##     family = binomial(link = "logit"), data = data_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.9402  -0.5720  -0.3817   0.5738   2.4811
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.049092   0.509904   7.941 2.01e-15 ***
## Sexmale     -2.944342   0.237537 -12.395  < 2e-16 ***
## Pclass2     -1.373954   0.311064  -4.417 1.00e-05 ***
## Pclass3     -2.510027   0.302105  -8.308  < 2e-16 ***
## Age         -0.026120   0.009663  -2.703  0.00687 **
## Child1       1.260669   0.536021   2.352  0.01868 *
```

```
## FamilySize  -0.237146   0.075534  -3.140  0.00169 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 889.27  on 667  degrees of freedom
## Residual deviance: 561.96  on 661  degrees of freedom
## AIC: 575.96
##
## Number of Fisher Scoring iterations: 5
```

Al igual que pasaba en el modelo2, todas las variables utilizadas han resultado significativas, y además el AIC del modelo ha mejorado ligeramente. Además la capacidad predictiva del modelo ha aumentado un poco:

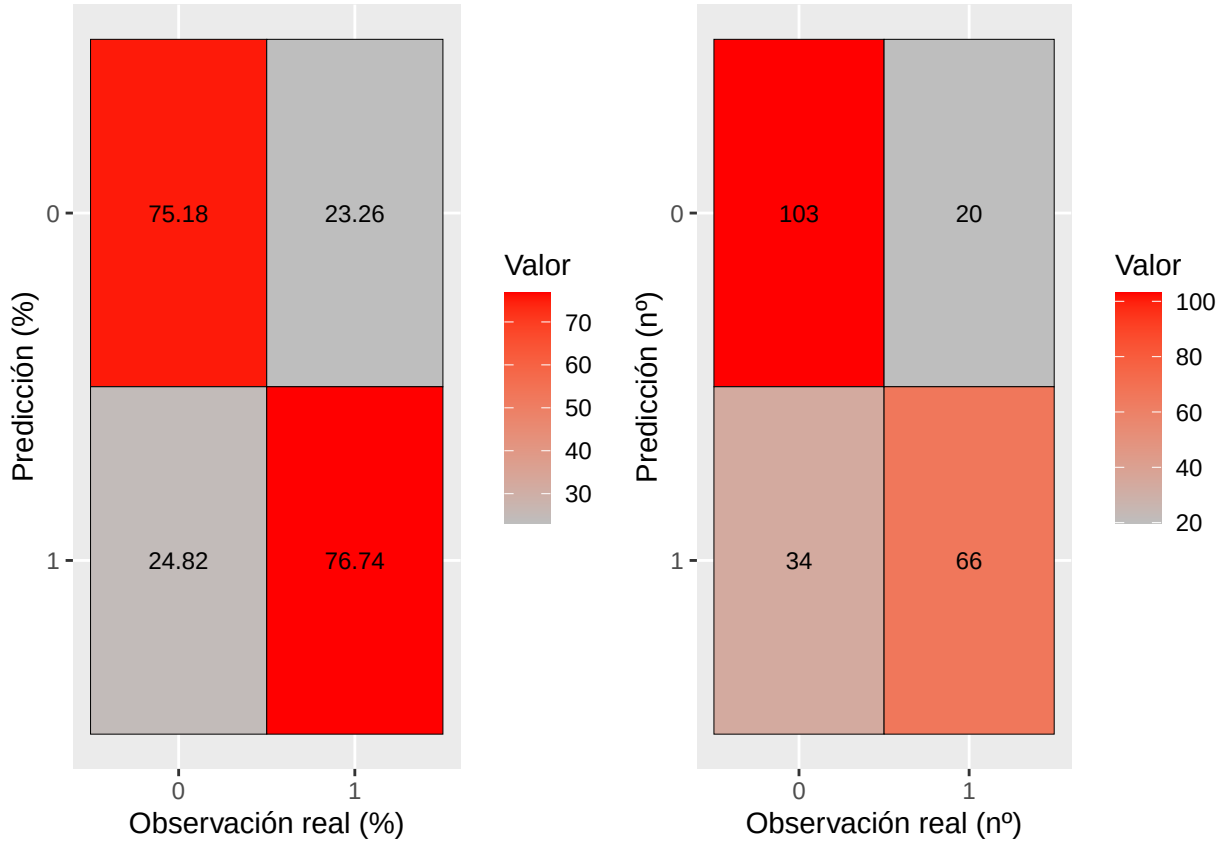
```
## Setting levels: control = 0, case = 1
```

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

```
##    predict_glm3
##       0   1
##   0 103  34
##   1  20  66
```

```
## [1] "El % de registros correctamente clasificados es: 75.7848 %"
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  223
##
##
##              | Prediction
##      Reality |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |       103 |        34 |       137 |
##              |     0.462 |     0.152 |           |
## -------------|-----------|-----------|-----------|
##            1 |        20 |        66 |        86 |
##              |     0.090 |     0.296 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       123 |       100 |       223 |
## -------------|-----------|-----------|-----------|
##
##
```



4.3.2.1.4 Cuarto modelo glm Antes de construir un cuarto modelo, vamos a analizar si existen variables independientes que afecten a la variable dependiente. Para ello primero definimos una variable 'objetivo' pero numérica

```
data$SurvivedNum <- as.integer(as.character(data$Survived))
```

Agrupamos el dataset por las variables Sex y Pclass

```
data.agrup <- data %>% group_by(Sex, Pclass)
```

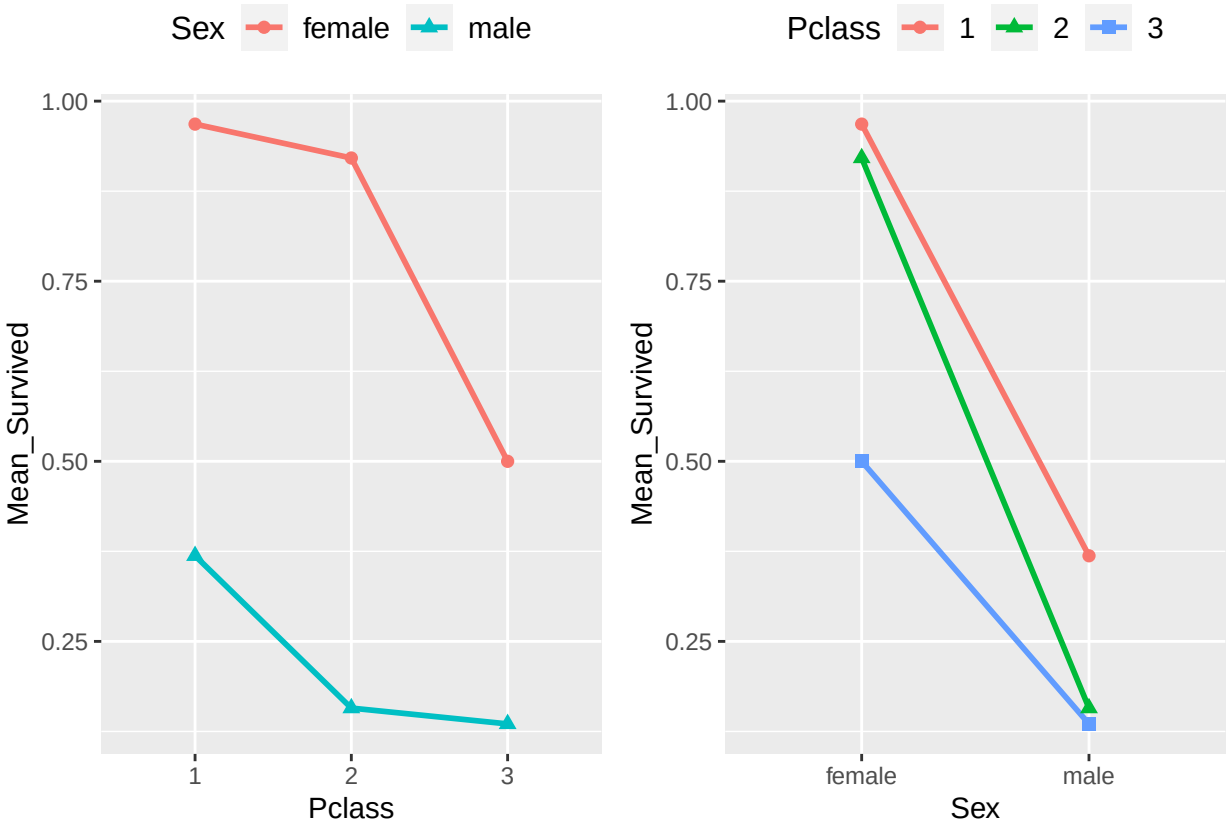
Calculamos la media creando un campo 'Mean_survived' de la media

```
data.mediasurvived <- summarize(.data = data.agrup, Mean_Survived = mean(SurvivedNum))
```

```
## `summarise()` regrouping output by 'Sex' (override with `.groups` argument)
```

```
kable(data.mediasurvived)
```

Sex	Pclass	Mean_Survived
female	1	0.9680851
female	2	0.9210526
female	3	0.5000000
male	1	0.3688525
male	2	0.1574074
male	3	0.1354467

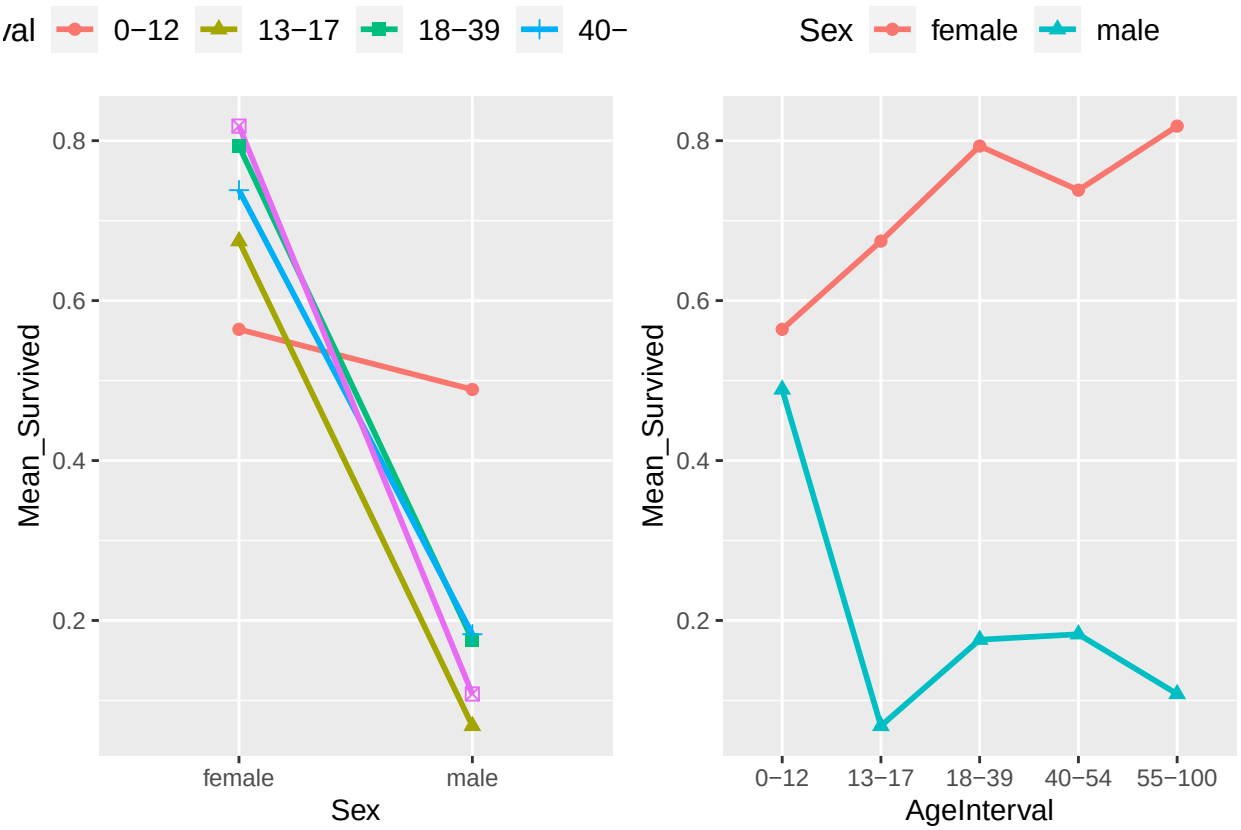


Tanto la variable **Sex** como la variable **Pclass** producen efecto en la variable **Mean_Survived**. También podemos comprobar cómo se produce interacción entre las variables **Sex** y **Pclass** respecto a **Mean_Survived**. Podemos ver cómo hay un descenso muy pronunciado cuando pasamos de mujeres a hombres y estamos tratando la clase 2ª. Observamos además cómo pasar a tercera clase afecta muy negativamente en el caso de las mujeres, ya que aunque no es una línea paralela, el hecho de estar en 1ª o 2ª clase no parece demasiado importante en el caso de las mujeres, pero al pasar a 3ª clase la caída de la media de supervivencia es muy importante.

Agrupamos el dataset por las variables **AgeInterval** y **Sex** y realizamos el mismo proceso

```
## `summarise()` regrouping output by 'AgeInterval' (override with `.groups` argument)
```

AgeInterval	Sex	Mean_Survived
0-12	female	0.5641026
0-12	male	0.4888889
13-17	female	0.6744186
13-17	male	0.0681818
18-39	female	0.7932961
18-39	male	0.1759777
40-54	female	0.7380952
40-54	male	0.1827957
55-100	female	0.8181818
55-100	male	0.1081081

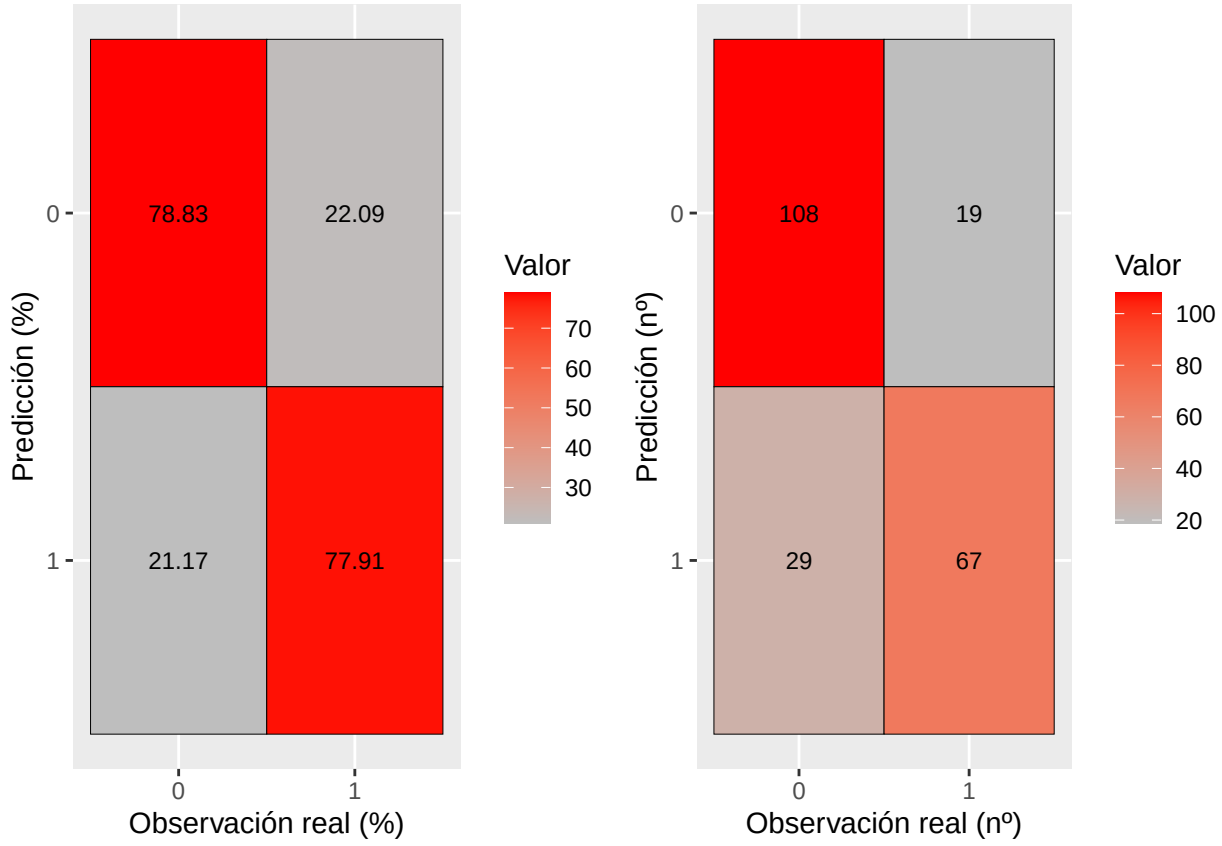


En este caso llama la atención el impacto de cuando pasamos el primer intervalo (menores o igualesa 12 años) al siguiente intervalo (de 13 a 17 años). La proporción de las mujeres aumenta mientras que en el caso de los hombres disminuye drásticamente. Además, no parece haber apenas diferencia en el primer tramo (los niños, se salvan con independencia del género)

Ahora construimos nuestro cuarto modelo de regresión logística. Utilizaremos algunas variables ya usadas en otros modelos más **AgeInterval**:**Sex**. También hemos añadido la interacción que generan **Sex**:**Pclass**. A estas variables le sumamos también las variables independientes **Sex**, **Parch** y **SibSp**.

```
##
## Call:
## glm(formula = Survived ~ Sex + Parch + SibSp + AgeInterval:Sex +
##     Sex:Pclass, family = binomial(link = "logit"), data = data_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8751  -0.5370  -0.3659   0.3086   2.7345
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                   4.47989    0.98996   4.525 6.03e-06 ***
## Sexmale                      -2.14960    1.10538  -1.945 0.051813 .
## Parch                        -0.03201    0.15333  -0.209 0.834659
## SibSp                        -0.29905    0.12172  -2.457 0.014012 *
## Sexfemale:AgeInterval13-17    0.07409    0.65940   0.112 0.910540
## Sexmale:AgeInterval13-17     -3.43817    0.89690  -3.833 0.000126 ***
## Sexfemale:AgeInterval18-39    0.15218    0.53883   0.282 0.777624
```

```
## Sexmale:AgeInterval18-39    -2.13013    0.51165  -4.163 3.14e-05 ***
## Sexfemale:AgeInterval40-54  -1.62083    0.81749  -1.983 0.047402 *
## Sexmale:AgeInterval40-54    -2.93654    0.62043  -4.733 2.21e-06 ***
## Sexfemale:AgeInterval55-100 -1.37122    1.69562  -0.809 0.418697
## Sexmale:AgeInterval55-100   -3.95106    0.83396  -4.738 2.16e-06 ***
## Sexfemale:Pclass2           -1.27210    0.93415  -1.362 0.173270
## Sexmale:Pclass2             -2.09381    0.44999  -4.653 3.27e-06 ***
## Sexfemale:Pclass3           -4.17990    0.88135  -4.743 2.11e-06 ***
## Sexmale:Pclass3             -2.06387    0.34622  -5.961 2.50e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 889.27  on 667  degrees of freedom
## Residual deviance: 519.90  on 652  degrees of freedom
## AIC: 551.9
##
## Number of Fisher Scoring iterations: 6

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

## Setting direction: controls < cases

##    predict_glm4
##       0   1
##   0 108  29
##   1  19  67

## [1] "El % de registros correctamente clasificados es: 78.4753 %"

##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  223
##
##
##              | Prediction
##      Reality |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |       108 |        29 |       137 |
##              |     0.484 |     0.130 |           |
## -------------|-----------|-----------|-----------|
##            1 |        19 |        67 |        86 |
##              |     0.085 |     0.300 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       127 |        96 |       223 |
## -------------|-----------|-----------|-----------|
##
##
```



4.3.2.1.5 Resumen modelos regresion Los diferentes modelos se han comportado de la siguiente forma:

```
knitr::kable(tabla.modelos)
```

Modelo	Precisión	Errores
glm1	79.372197309417	46
glm2	78.4753363228699	48
glm3	75.7847533632287	54
glm4	78.4753363228699	48

Curvas ROC

```
r1$auc
```

```
## Area under the curve: 0.799
```

```
r2$auc
```

```
## Area under the curve: 0.8055
```

```
r3$auc
```

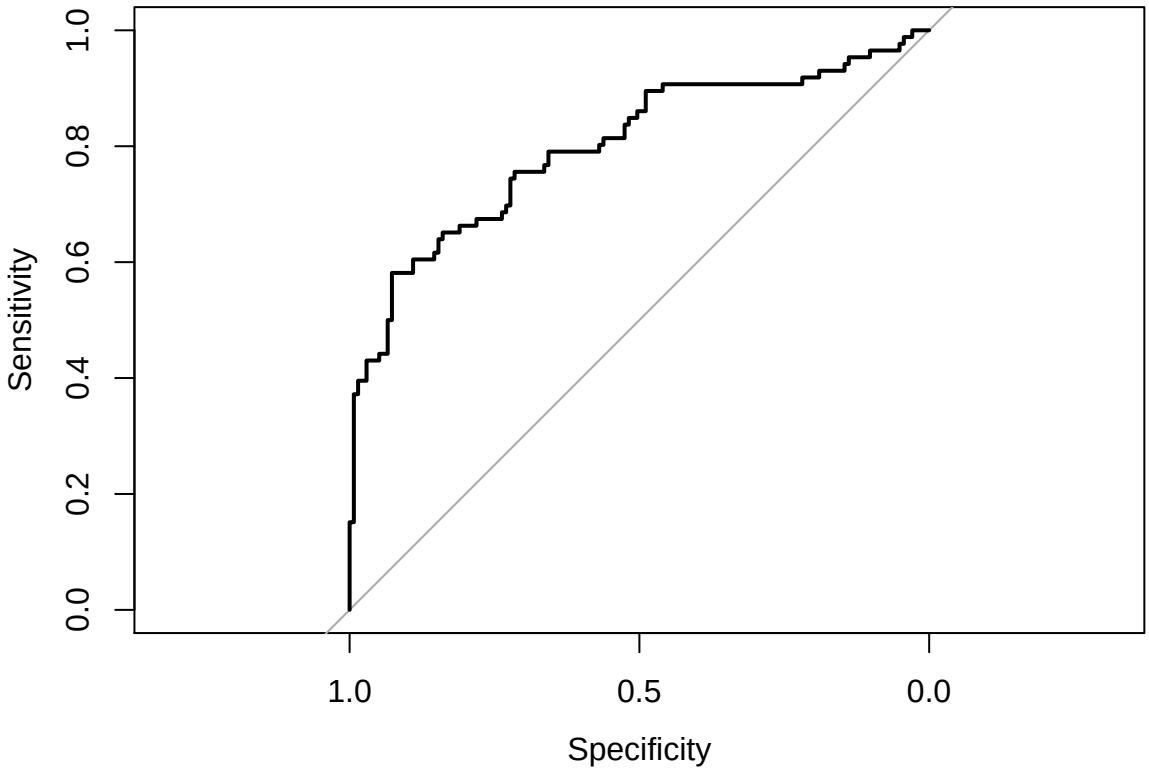
```
## Area under the curve: 0.8116
```

```
r4$auc
```

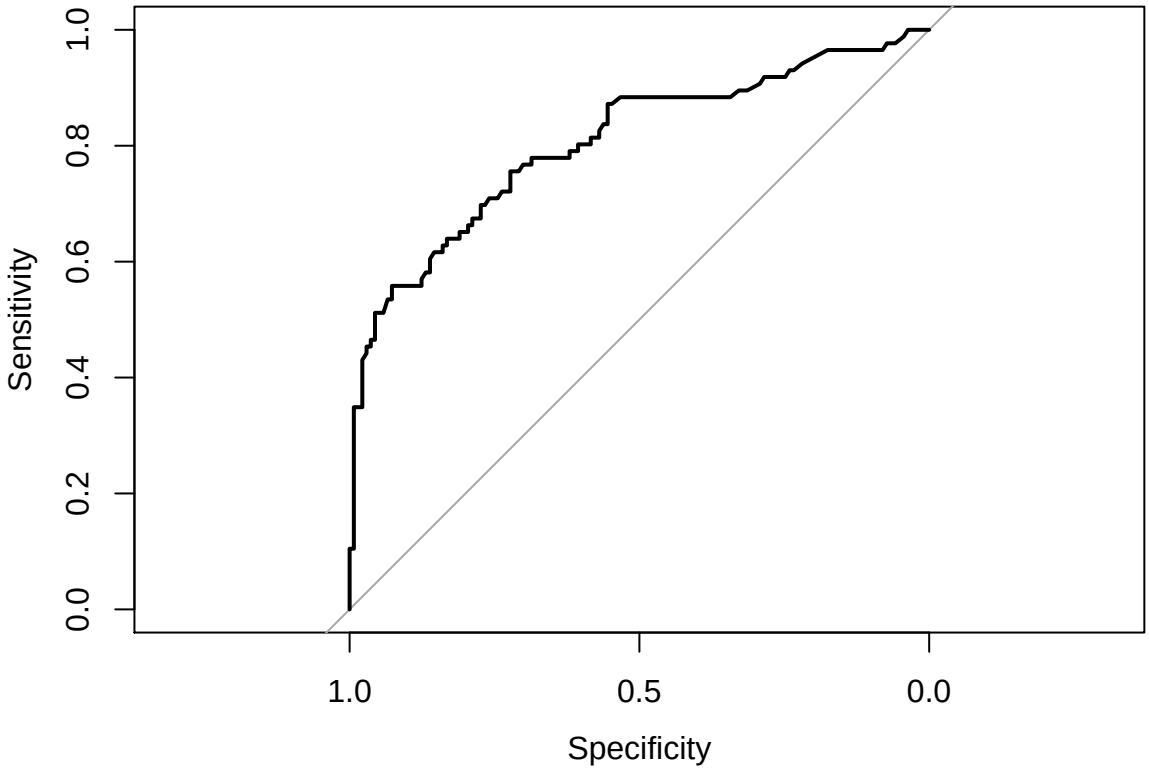
```
## Area under the curve: 0.8289
```

```
data$SurvivedNum <- NULL
```

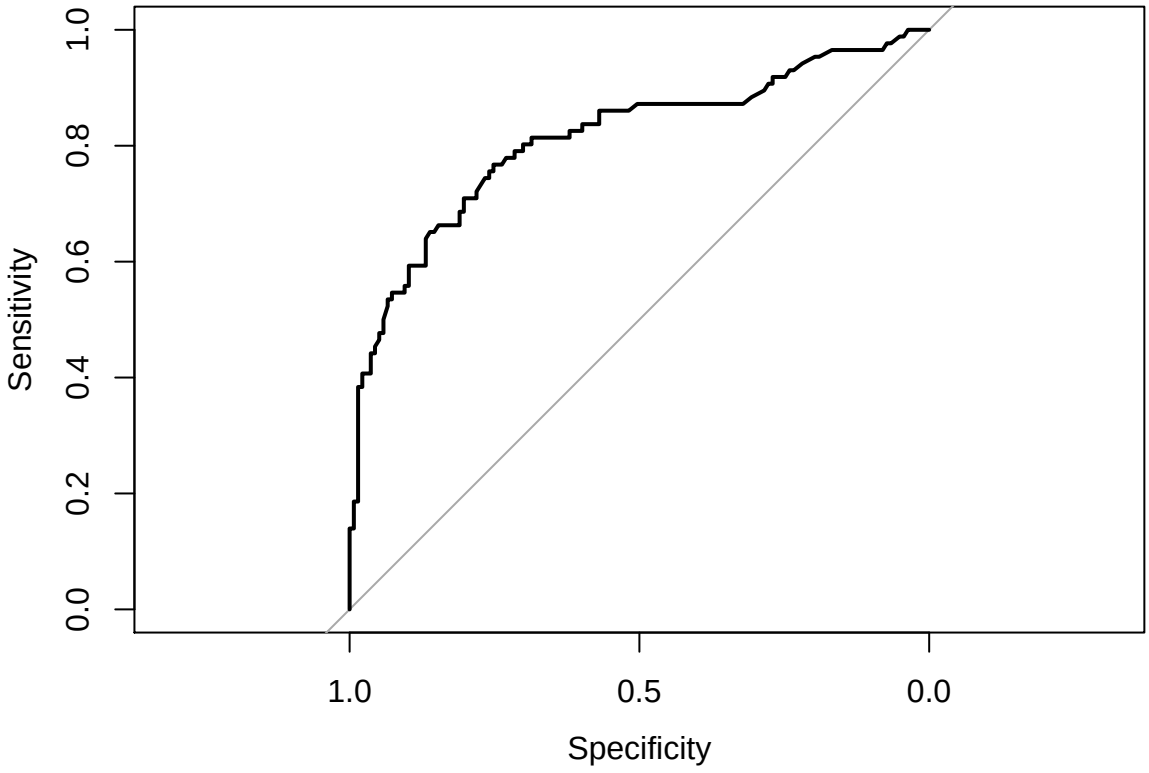
```
plot.roc(r1)
```



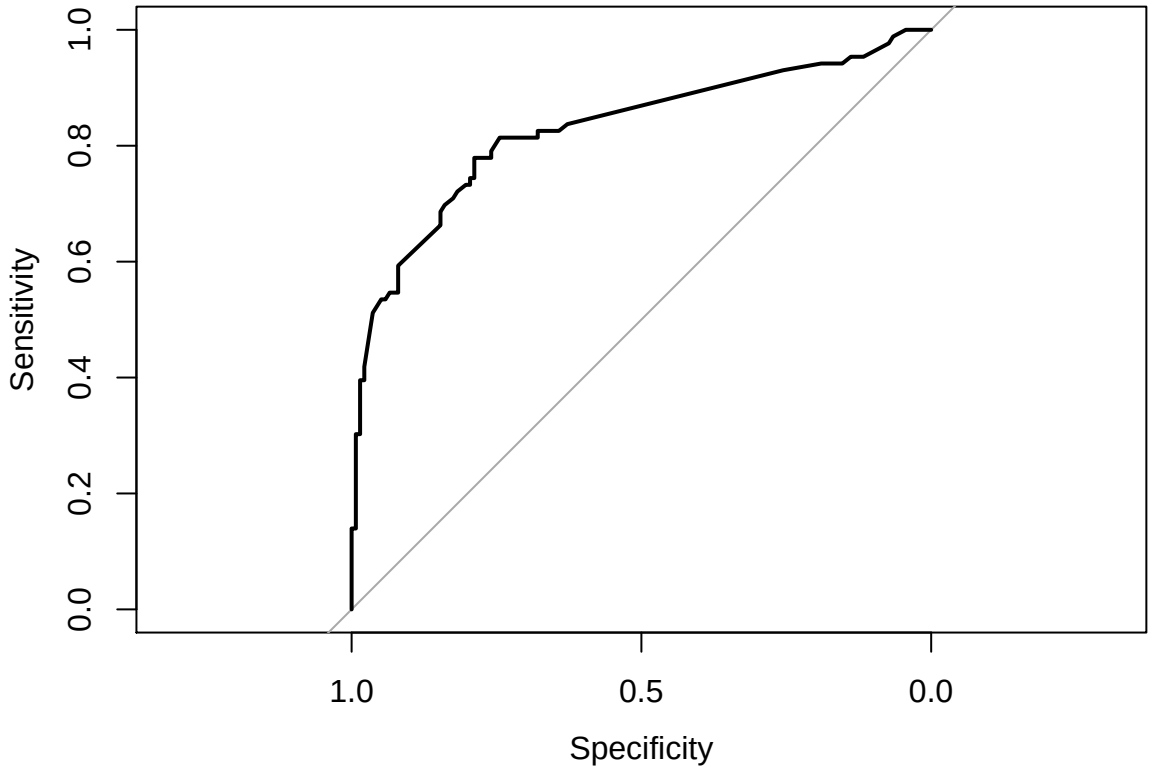

```
plot.roc(r2)
```




```
plot.roc(r3)
```




```
plot.roc(r4)
```



TODO: Pintarlas en el mismo gráfico y hacer una explicación TODO: ¿Explicación breve de las curvas?

4.3.2.2 Árboles de decisión

Los árboles de decisión son modelos muy sencillos de construir.

4.3.2.2.1 Primer árbol de decisión

Creamos un primer árbol con los parámetros básicos

```
tctrl <- caret::trainControl(method = "repeatedcv",
                             number=10, repeats = 3)
model_tree1 <- caret::train(Survived~Sex+FamilySize+Child+Age+Pclass,
                            data=data_train,
                            method="rpart",
                            trControl = tctrl)
summary(model_tree1)
```

```
## Call:
## (function (formula, data, weights, subset, na.action = na.rpart,
##     method, model = FALSE, x = FALSE, y = TRUE, parms, control,
##     cost, ...)
## {
##     Call <- match.call()
##     if (is.data.frame(model)) {
##         m <- model
##         model <- FALSE
##     }
##     else {
##         indx <- match(c("formula", "data", "weights", "subset"),
##             names(Call), nomatch = 0)
```

```
##         if (indx[1] == 0)
##             stop("a 'formula' argument is required")
##         temp <- Call[c(1, indx)]
##         temp$na.action <- na.action
##         temp[[1]] <- quote(stats::model.frame)
##         m <- eval.parent(temp)
##     }
##     Terms <- attr(m, "terms")
##     if (any(attr(Terms, "order") > 1))
##         stop("Trees cannot handle interaction terms")
##     Y <- model.response(m)
##     wt <- model.weights(m)
##     if (any(wt < 0))
##         stop("negative weights not allowed")
##     if (!length(wt))
##         wt <- rep(1, nrow(m))
##     offset <- model.offset(m)
##     X <- rpart.matrix(m)
##     nobs <- nrow(X)
##     nvar <- ncol(X)
##     if (missing(method)) {
##         method <- if (is.factor(Y) || is.character(Y))
##             "class"
##         else if (inherits(Y, "Surv"))
##             "exp"
##         else if (is.matrix(Y))
##             "poisson"
##         else "anova"
##     }
##     if (is.list(method)) {
##         mlist <- method
##         method <- "user"
##         init <- if (missing(parms))
##             mlist$init(Y, offset, wt = wt)
##         else mlist$init(Y, offset, parms, wt)
##         keep <- rpartcallback(mlist, nobs, init)
##         method.int <- 4
##         parms <- init$parms
##     }
##     else {
##         method.int <- pmatch(method, c("anova", "poisson", "class",
##             "exp"))
##         if (is.na(method.int))
##             stop("Invalid method")
##         method <- c("anova", "poisson", "class", "exp")[method.int]
##         if (method.int == 4)
##             method.int <- 2
##         init <- if (missing(parms))
##             get(paste("rpart", method, sep = "."), envir = environment())(Y,
##                 offset, , wt)
##         else get(paste("rpart", method, sep = "."), envir = environment())(Y,
##             offset, parms, wt)
##         ns <- asNamespace("rpart")
##         if (!is.null(init$print))
```

```
##             environment(init$print) <- ns
##         if (!is.null(init$summary))
##             environment(init$summary) <- ns
##         if (!is.null(init$text))
##             environment(init$text) <- ns
##     }
##     Y <- init$y
##     xlevels <- .getXlevels(Terms, m)
##     cats <- rep(0, ncol(X))
##     if (!is.null(xlevels))
##         cats[match(names(xlevels), colnames(X))] <- unlist(lapply(xlevels,
##             length))
##     extraArgs <- list(...)
##     if (length(extraArgs)) {
##         controlargs <- names(formals(rpart.control))
##         indx <- match(names(extraArgs), controlargs, nomatch = 0)
##         if (any(indx == 0))
##             stop(gettextf("Argument %s not matched", names(extraArgs)[indx ==
##                 0]), domain = NA)
##     }
##     controls <- rpart.control(...)
##     if (!missing(control))
##         controls[names(control)] <- control
##     xval <- controls$xval
##     if (is.null(xval) || (length(xval) == 1 && xval == 0) ||
##         method == "user") {
##         xgroups <- 0
##         xval <- 0
##     }
##     else if (length(xval) == 1) {
##         xgroups <- sample(rep(1:xval, length = nobs), nobs, replace = FALSE)
##     }
##     else if (length(xval) == nobs) {
##         xgroups <- xval
##         xval <- length(unique(xgroups))
##     }
##     else {
##         if (!is.null(attr(m, "na.action"))) {
##             temp <- as.integer(attr(m, "na.action"))
##             xval <- xval[-temp]
##             if (length(xval) == nobs) {
##                 xgroups <- xval
##                 xval <- length(unique(xgroups))
##             }
##             else stop("Wrong length for 'xval'")
##         }
##         else stop("Wrong length for 'xval'")
##     }
##     if (missing(cost))
##         cost <- rep(1, nvar)
##     else {
##         if (length(cost) != nvar)
##             stop("Cost vector is the wrong length")
##         if (any(cost <= 0))
```

```
##             stop("Cost vector must be positive")
##     }
##     tfun <- function(x) if (is.matrix(x))
##         rep(is.ordered(x), ncol(x))
##     else is.ordered(x)
##     labs <- sub("^`(.*)`$", "\\1", attr(Terms, "term.labels"))
##     isord <- unlist(lapply(m[labs], tfun))
##     storage.mode(X) <- "double"
##     storage.mode(wt) <- "double"
##     temp <- as.double(unlist(init$parms))
##     if (!length(temp))
##         temp <- 0
##     rpfit <- .Call(C_rpart, ncat = as.integer(cats * !isord),
##         method = as.integer(method.int), as.double(unlist(controls)),
##         temp, as.integer(xval), as.integer(xgroups), as.double(t(init$y)),
##         X, wt, as.integer(init$numy), as.double(cost))
##     nsplit <- nrow(rpfit$isplit)
##     ncat <- if (!is.null(rpfit$csplit))
##         nrow(rpfit$csplit)
##     else 0
##     if (nsplit == 0)
##         xval <- 0
##     numcp <- ncol(rpfit$cptable)
##     temp <- if (nrow(rpfit$cptable) == 3)
##         c("CP", "nsplit", "rel error")
##     else c("CP", "nsplit", "rel error", "xerror", "xstd")
##     dimnames(rpfit$cptable) <- list(temp, 1:numcp)
##     tname <- c("<leaf>", colnames(X))
##     splits <- matrix(c(rpfit$isplit[, 2:3], rpfit$dsplit), ncol = 5,
##         dimnames = list(tname[rpfit$isplit[, 1] + 1], c("count",
##             "ncat", "improve", "index", "adj")))
##     index <- rpfit$inode[, 2]
##     nadd <- sum(isord[rpfit$isplit[, 1]])
##     if (nadd > 0) {
##         newc <- matrix(0, nadd, max(cats))
##         cvar <- rpfit$isplit[, 1]
##         indx <- isord[cvar]
##         cdir <- splits[indx, 2]
##         ccut <- floor(splits[indx, 4])
##         splits[indx, 2] <- cats[cvar[indx]]
##         splits[indx, 4] <- ncat + 1:nadd
##         for (i in 1:nadd) {
##             newc[i, 1:(cats[(cvar[indx])[i]])] <- -as.integer(cdir[i])
##             newc[i, 1:ccut[i]] <- as.integer(cdir[i])
##         }
##         catmat <- if (ncat == 0)
##             newc
##         else {
##             cs <- rpfit$csplit
##             ncs <- ncol(cs)
##             ncc <- ncol(newc)
##             if (ncs < ncc)
##                 cs <- cbind(cs, matrix(0, nrow(cs), ncc - ncs))
##             rbind(cs, newc)
```

```
##         }
##         ncat <- ncat + nadd
##     }
##     else catmat <- rpfit$csplit
##     if (nsplit == 0) {
##         frame <- data.frame(row.names = 1, var = "<leaf>", n = rpfit$inode[,
##             5], wt = rpfit$dnode[, 3], dev = rpfit$dnode[, 1],
##             yval = rpfit$dnode[, 4], complexity = rpfit$dnode[,
##                 2], ncompete = 0, nsurrogate = 0)
##     }
##     else {
##         temp <- ifelse(index == 0, 1, index)
##         svar <- ifelse(index == 0, 0, rpfit$isplit[temp, 1])
##         frame <- data.frame(row.names = rpfit$inode[, 1], var = tname[svar +
##             1], n = rpfit$inode[, 5], wt = rpfit$dnode[, 3],
##             dev = rpfit$dnode[, 1], yval = rpfit$dnode[, 4],
##             complexity = rpfit$dnode[, 2], ncompete = pmax(0,
##                 rpfit$inode[, 3] - 1), nsurrogate = rpfit$inode[,
##                 4])
##     }
##     if (method.int == 3) {
##         numclass <- init$numresp - 2
##         nodeprob <- rpfit$dnode[, numclass + 5]/sum(wt)
##         temp <- pmax(1, init$counts)
##         temp <- rpfit$dnode[, 4 + (1:numclass)] %*% diag(init$parms$prior/temp)
##         yprob <- temp/rowSums(temp)
##         yval2 <- matrix(rpfit$dnode[, 4 + (0:numclass)], ncol = numclass +
##             1)
##         frame$yval2 <- cbind(yval2, yprob, nodeprob)
##     }
##     else if (init$numresp > 1)
##         frame$yval2 <- rpfit$dnode[, -(1:3), drop = FALSE]
##     if (is.null(init$summary))
##         stop("Initialization routine is missing the 'summary' function")
##     functions <- if (is.null(init$print))
##         list(summary = init$summary)
##     else list(summary = init$summary, print = init$print)
##     if (!is.null(init$text))
##         functions <- c(functions, list(text = init$text))
##     if (method == "user")
##         functions <- c(functions, mlist)
##     where <- rpfit$which
##     names(where) <- row.names(m)
##     ans <- list(frame = frame, where = where, call = Call, terms = Terms,
##         cptable = t(rpfit$cptable), method = method, parms = init$parms,
##         control = controls, functions = functions, numresp = init$numresp)
##     if (nsplit)
##         ans$splits = splits
##     if (ncat > 0)
##         ans$csplit <- catmat + 2
##     if (nsplit)
##         ans$variable.importance <- importance(ans)
##     if (model) {
##         ans$model <- m
```

```
##         if (missing(y))
##             y <- FALSE
##     }
##     if (y)
##         ans$y <- Y
##     if (x) {
##         ans$x <- X
##         ans$wt <- wt
##     }
##     ans$ordered <- isord
##     if (!is.null(attr(m, "na.action")))
##         ans$na.action <- attr(m, "na.action")
##     if (!is.null(xlevels))
##         attr(ans, "xlevels") <- xlevels
##     if (method == "class")
##         attr(ans, "ylevels") <- init$ylevels
##     class(ans) <- "rpart"
##     ans
## })(formula = .outcome ~ ., data = list(c(0, 0, 0, 1, 0, 1, 1,
## 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1,
## 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1,
## 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1,
## 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0,
## 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0,
## 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0,
## 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1,
## 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1,
## 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1,
## 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1,
## 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0,
## 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0,
## 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0,
## 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1,
## 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1,
## 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1,
## 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0,
## 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1,
## 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1,
## 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1,
## 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1,
## 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1,
## 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0,
## 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,
## 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1,
## 1, 1, 0, 1, 0, 1, 0, 1, 1, 0), c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
```

```
## 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
## 2, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
## 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
## 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 6, 3, 3, 5,
## 5, 6, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 2, 2, 2, 2, 3, 3, 3, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
## 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
## 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 5, 3, 3, 4, 4, 4, 4, 6, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 5, 6, 2,
## 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
## 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3,
## 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 5, 5, 6, 6, 7, 7,
## 8, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 5, 5, 4, 5, 5, 5, 5, 5, 6,
## 6, 6, 6, 6, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7, 7, 7, 7, 8, 8, 8, 8,
## 11, 11, 11, 11, 11, 11, 11), c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0,
## 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
## 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0,
## 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0,
## 0, 0, 0, 0, 0, 1, 0), c(30, 33, 16, 47, 22, 35, 33, 19, 21, 40,
## 40, 54, 45.5, 45, 18, 28, 29, 38, 30, 45, 29, 62, 62, 38, 64,
## 58, 42, 31, 47, 46, 50, 36, 56, 45, 52, 55, 28, 56, 34, 31, 60,
## 11, 71, 27, 62, 24, 42, 41, 32, 36, 24, 35, 30, 54, 40, 32, 48,
## 65, 24, 28, 56, 26, 32, 21, 24, 49, 24, 48, 61, 47, 44, 65, 58,
## 36, 35, 42, 35, 38, 27, 18, 38, 65, 58, 54, 16, 21, 22, 30, 36,
## 44, 15, 51, 24, 43, 50, 31, 36, 22, 58, 17, 27, 4, 19, 48, 35,
## 35, 31, 49, 33, 31, 27, 54, 50, 37, 49, 54, 35, 35, 45, 71, 35,
## 50, 46, 33, 18, 50, 25, 17, 42, 48, 25, 38, 27, 19, 60, 36, 38,
## 49, 52, 48, 18, 37, 70, 45, 49, 40, 60, 39, 47, 36, 14, 2, 36,
## 64, 50, 53, 18, 21, 24, 50, 34, 52, 18, 40, 40, 30, 28, 27, 27,
## 55, 30, 42, 27, 30, 42, 25, 30, 47, 34, 30, 24, 70, 28, 16, 36,
## 22, 30, 35, 34, 57, 27, 50, 19, 28, 19, 27, 26, 32, 48, 66, 21,
## 25, 23, 59, 45, 23, 28, 36, 54, 28, 31, 23, 60, 24, 24, 19, 39,
## 36, 31, 27, 32, 50, 34, 18, 36, 62, 38, 32.5, 39, 16, 19, 28,
## 21, 46, 54, 51, 29, 35, 24, 17, 28, 35, 25, 34, 28, 50, 7, 36.5,
## 18, 8, 24, 42, 27, 25, 29, 34, 32.5, 54, 29, 28, 29, 34, 27,
## 29, 19, 44, 21, 24, 36, 44, 30, 2, 45, 3, 43, 34, 40, 8, 3, 25,
## 31, 31, 31, 22, 2, 4, 24, 3, 36, 22, 48, 25, 33, 5, 54, 32, 21,
## 24, 1, 23, 4, 24, 36, 28, 24, 28, 17, 51, 17, 27, 24, 36, 32,
## 30, 19, 28, 21, 34, 47, 38, 30, 26, 18, 45.5, 18, 30, 20, 26,
## 30.5, 47, 51, 28, 29, 26, 26, 24, 22, 30.5, 50, 21, 20, 28, 26,
## 30, 17, 65, 41, 31, 20, 32, 28, 20, 21, 35, 16, 29, 42, 14, 23,
## 21, 32, 28, 47, 26, 13, 40, 31, 22, 34, 17, 24, 21, 29, 15, 18,
## 38, 17, 19, 21, 20, 18, 49, 56, 18, 43, 27, 28, 18, 25, 22, 22,
## 45.5, 26, 25, 55, 20, 17, 21, 61, 32, 29, 35, 19, 33, 38, 17,
## 28, 35, 32, 20, 22, 24, 15, 25, 35, 17, 11, 21, 17, 19, 21, 20.5,
## 33, 26, 27, 61, 22, 35, 5, 30, 47, 22, 25, 26, 45, 19, 35, 22,
## 33, 24, 23.5, 30, 45, 28.5, 51, 29, 27, 32, 16, 19, 44, 32, 18,
## 59, 26, 4, 22, 21, 32, 27, 20, 35, 45, 24, 54, 18, 19, 28, 20,
## 30, 25, 28, 20, 36, 48, 31, 16, 26, 25, 39, 16, 21, 13, 21, 20,
## 0.42, 19, 37, 43, 18, 25, 21, 51, 18, 22, 1, 29, 32, 36, 35,
## 33, 29, 20, 21, 32, 35, 32, 21, 22, 40.5, 30.5, 23, 22, 23, 47,
## 58, 22, 30, 48, 42, 21, 28.5, 42, 20, 26, 19, 20.5, 23.5, 70.5,
## 26, 29, 20, 36, 38, 55.5, 41, 44, 27, 18, 42, 2, 18, 41, 10,
## 27, 24, 40.5, 58, 31, 29, 19, 1, 24, 29, 41, 19, 24, 12, 44,
## 38, 36, 30, 25, 18, 25, 32, 4, 29, 47, 19, 25, 21, 22, 27, 21,
## 24, 19, 31, 26, 26, 20, 15, 24, 35, 20, 40, 23, 36, 17, 15, 33,
## 4, 4, 16, 18, 34, 18, 35, 20, 30, 30, 36, 9, 9, 40, 32, 29, 35,
## 26, 17, 1, 18, 26, 48, 16, 45, 40, 39, 38, 43, 18, 18, 28, 27,
## 16, 41, 24, 37, 32, 5, 0.75, 21, 9, 33, 15, 2, 51, 8, 3, 9, 10,
## 2, 7, 7, 2, 14, 16, 8, 1, 3, 5, 2, 17, 4, 11, 9, 6, 1, 14, 11,
## 16, 41, 9, 23, 22, 28, 2, 18), c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0), c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
```

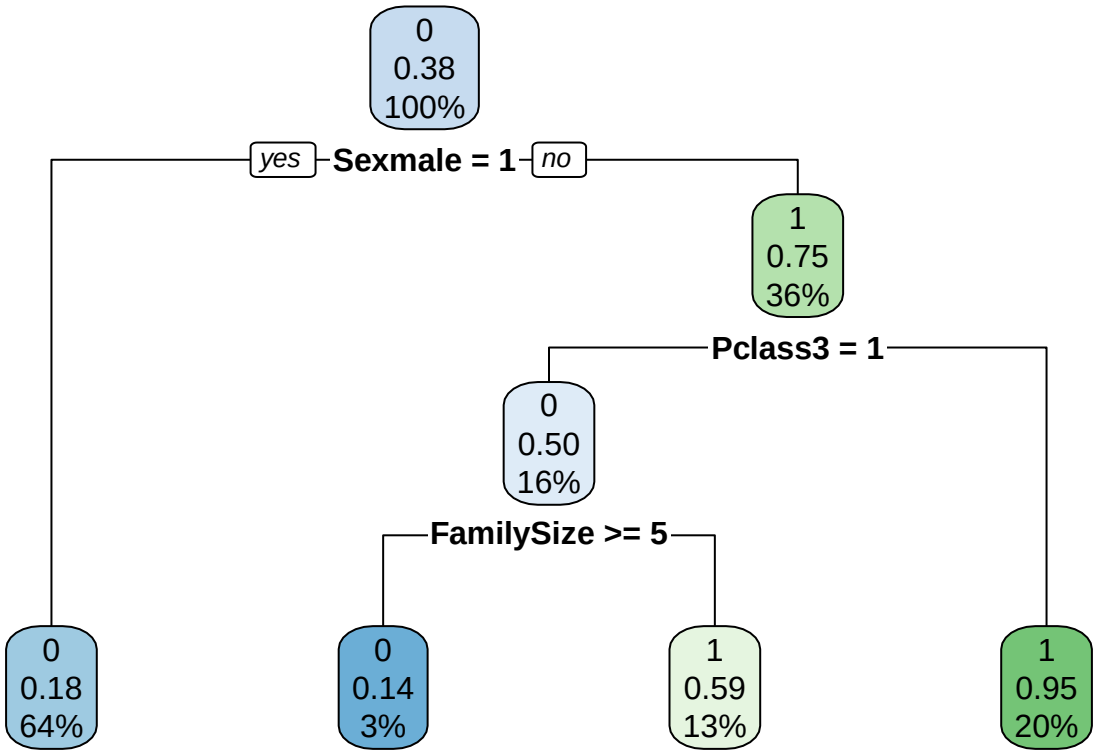
```
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1), c(2, 2, 2, 1, 2, 2, 1, 2, 1, 1, 2, 1, 1, 1, 1, 1,
## 1, 2, 2, 1, 2, 1, 2, 2, 1, 1, 2, 1, 1, 1, 1, 1, 2, 1, 2, 1, 2,
## 1, 2, 1, 1, 1, 1, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 1,
## 1, 2, 1, 2, 2, 2, 2, 2, 2, 2, 1, 1, 2, 1, 2, 2, 2, 2, 2, 1, 2,
## 1, 1, 1, 2, 1, 2, 1, 2, 2, 2, 2, 2, 1, 1, 2, 2, 2, 2, 2, 1, 2,
## 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 2, 2, 2, 2, 1, 2, 2,
## 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 1, 2,
## 1, 2, 1, 2, 2, 2, 2, 2, 2, 1, 2, 1, 2, 2, 2, 2, 2, 2, 1, 1, 1,
## 2, 2, 2, 2, 1, 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1,
## 2, 1, 1, 2, 2, 1, 1, 2, 2, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 2,
## 1, 1, 1, 1, 1, 2, 1, 1, 2, 1, 2, 1, 1, 1, 2, 2, 1, 2, 1, 2, 2,
## 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 2, 2, 1, 2, 2, 1, 2, 2,
## 1, 2, 2, 2, 1, 2, 1, 1, 1, 1, 1, 2, 2, 2, 1, 1, 1, 2, 1, 1, 2,
## 2, 1, 1, 2, 2, 2, 1, 2, 2, 2, 2, 2, 1, 2, 1, 2, 2, 2, 2, 2, 1,
## 2, 2, 1, 2, 2, 2, 1, 1, 2, 2, 1, 2, 2, 1, 1, 1, 1, 2, 1, 1, 2,
## 2, 1, 2, 1, 2, 1, 1, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 2, 2, 1, 1,
## 1, 2, 1, 1, 1, 2, 1, 2, 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1,
## 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 2, 1,
## 1, 2, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 2, 2, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 2, 1, 2, 2, 1, 1, 1, 1, 2, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1,
## 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 2, 1, 2, 1, 1, 1, 1, 1, 2, 1,
## 1, 1, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 2,
## 1, 1, 1, 2, 2, 1, 1, 1, 2, 2, 2, 2, 1, 1, 2, 2, 2, 1, 2, 2, 1,
## 1, 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 2, 1,
## 1, 2, 1, 2, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 2, 2, 1, 2, 1, 1, 2,
## 1, 1, 1, 2, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1,
## 1, 1, 1, 2, 2, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 2, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1)), control = list(20, 7, 0, 4, 5, 2, 0, 30, 0))
##   n= 668
##
##           CP nsplit rel error
## 1 0.47265625      0 1.0000000
## 2 0.03125000      1 0.5273438
## 3 0.01757812      3 0.4648438
##
## Variable importance
##    Sexmale    Pclass3 FamilySize        Age    Pclass2     Child1
##         63         15         10          5          5          1
##
## Node number 1: 668 observations,    complexity param=0.4726562
##   predicted class=0  expected loss=0.3832335  P(node) =1
##     class counts:   412   256
##    probabilities: 0.617 0.383
##   left son=2 (427 obs) right son=3 (241 obs)
##   Primary splits:
```

```
##       Sexmale    < 0.5  to the right, improve=102.006600, (0 missing)
##       Pclass3    < 0.5  to the right, improve= 35.590460, (0 missing)
##       FamilySize < 1.5  to the left,  improve= 12.830360, (0 missing)
##       Age        < 5.5  to the right, improve=  4.986881, (0 missing)
##       Child1     < 0.5  to the left,  improve=  4.028764, (0 missing)
##   Surrogate splits:
##       FamilySize < 1.5  to the left,  agree=0.659, adj=0.054, (0 split)
##       Age        < 15.5 to the right, agree=0.644, adj=0.012, (0 split)
##
## Node number 2: 427 observations
##   predicted class=0  expected loss=0.175644  P(node) =0.6392216
##     class counts:   352    75
##    probabilities: 0.824 0.176
##
## Node number 3: 241 observations,    complexity param=0.03125
##   predicted class=1  expected loss=0.2489627  P(node) =0.3607784
##     class counts:    60   181
##    probabilities: 0.249 0.751
##   left son=6 (108 obs) right son=7 (133 obs)
##   Primary splits:
##       Pclass3    < 0.5  to the right, improve=24.66583000, (0 missing)
##       FamilySize < 4.5  to the right, improve=12.57550000, (0 missing)
##       Pclass2    < 0.5  to the left,  improve= 5.30974800, (0 missing)
##       Age        < 18.5 to the left,  improve= 2.80259000, (0 missing)
##       Child1     < 0.5  to the right, improve= 0.06214367, (0 missing)
##   Surrogate splits:
##       Pclass2    < 0.5  to the left,  agree=0.697, adj=0.324, (0 split)
##       Age        < 23.5 to the left,  agree=0.676, adj=0.278, (0 split)
##       FamilySize < 4.5  to the right, agree=0.622, adj=0.157, (0 split)
##       Child1     < 0.5  to the right, agree=0.573, adj=0.046, (0 split)
##
## Node number 6: 108 observations,    complexity param=0.03125
##   predicted class=0  expected loss=0.5  P(node) =0.1616766
##     class counts:    54    54
##    probabilities: 0.500 0.500
##   left son=12 (22 obs) right son=13 (86 obs)
##   Primary splits:
##       FamilySize < 4.5  to the right, improve=7.3065540, (0 missing)
##       Age        < 39   to the right, improve=4.9090910, (0 missing)
##       Child1     < 0.5  to the left,  improve=0.3935223, (0 missing)
##   Surrogate splits:
##       Age < 40.5 to the right, agree=0.815, adj=0.091, (0 split)
##
## Node number 7: 133 observations
##   predicted class=1  expected loss=0.04511278  P(node) =0.1991018
##     class counts:     6   127
##    probabilities: 0.045 0.955
##
## Node number 12: 22 observations
##   predicted class=0  expected loss=0.1363636  P(node) =0.03293413
##     class counts:    19     3
##    probabilities: 0.864 0.136
##
## Node number 13: 86 observations
```

```
##   predicted class=1  expected loss=0.4069767  P(node) =0.1287425
##     class counts:    35    51
##    probabilities: 0.407 0.593
```

Dibujamos el árbol

```
rpart.plot(model_tree1$finalModel)
```



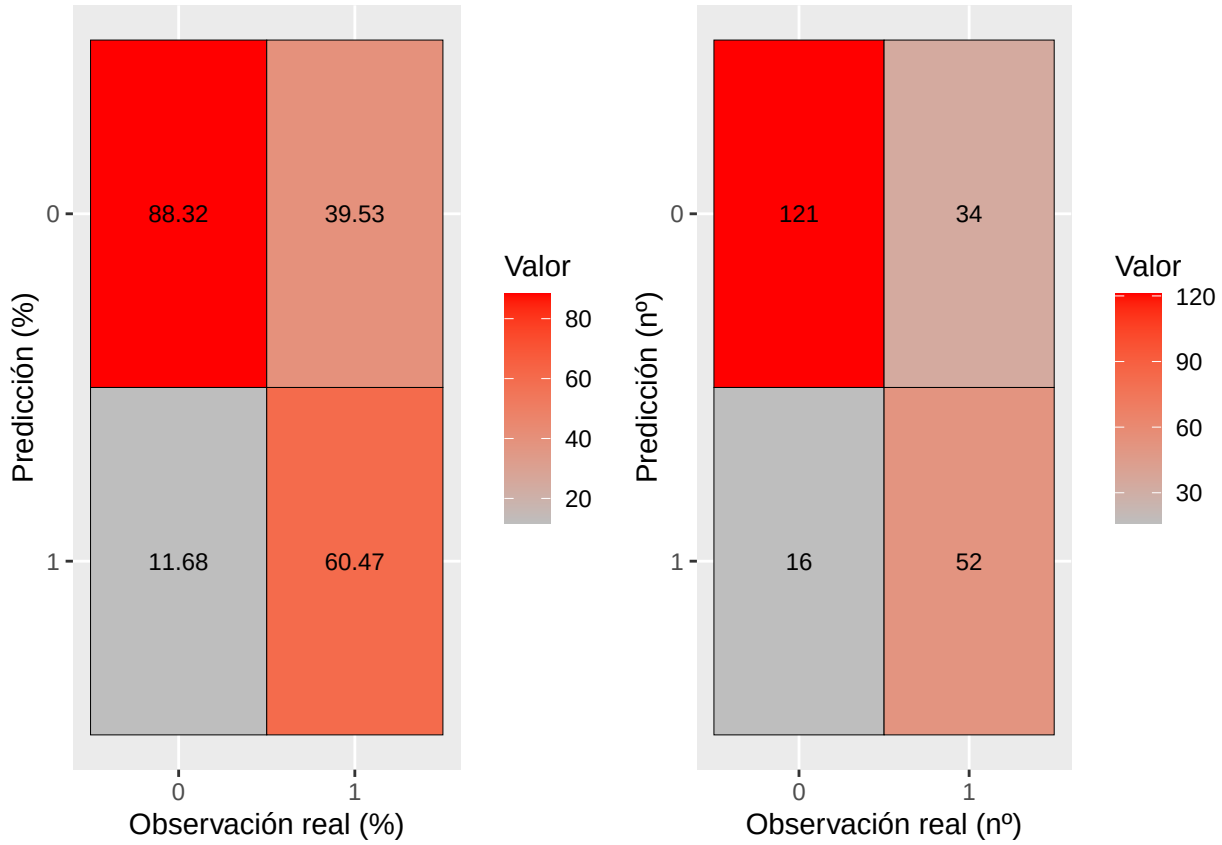
Realizamos la predicción con el modelo construido con rpart y el dataset test

```
##    predict_tree1
##       0   1
##   0 121  16
##   1  34  52
```

```
## [1] "El % de registros correctamente clasificados es: 77.5785 %"
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  223
##
##
```

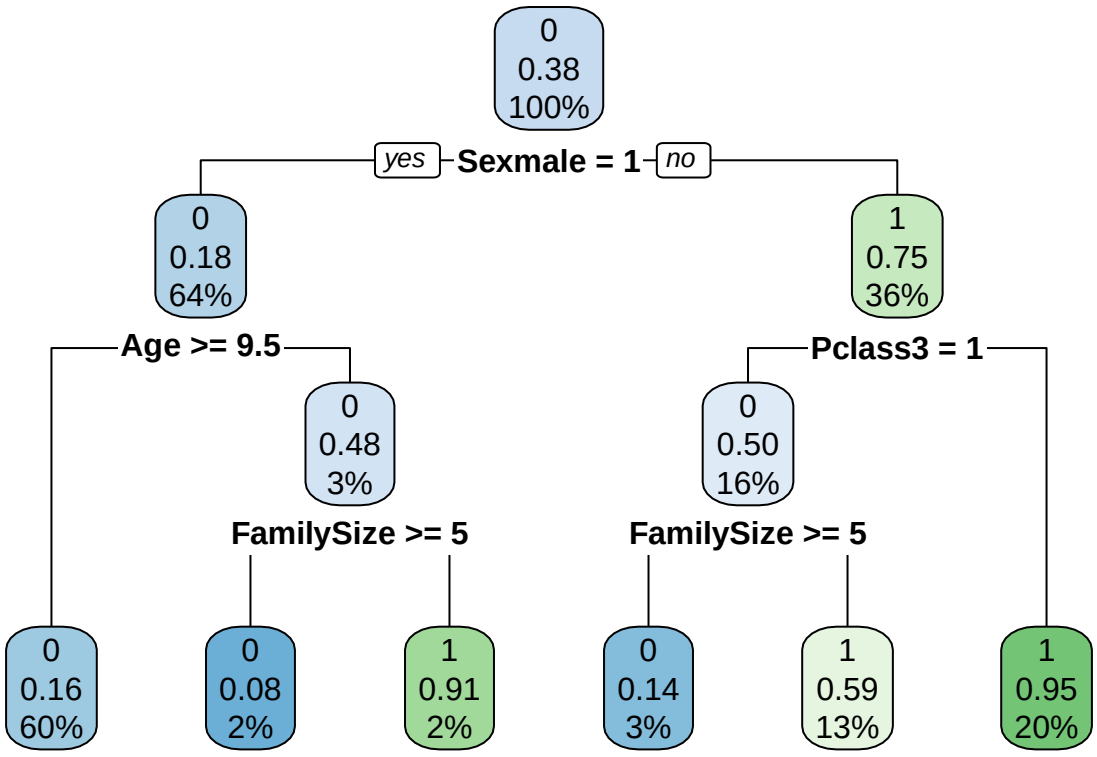
```
##              | Prediction
##      Reality |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |       121 |        16 |       137 |
##              |     0.543 |     0.072 |           |
## -------------|-----------|-----------|-----------|
##            1 |        34 |        52 |        86 |
##              |     0.152 |     0.233 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       155 |        68 |       223 |
## -------------|-----------|-----------|-----------|
##
##
```



4.3.2.2.2 Segundo árbol de decisión Manteniendo las mismas variables predictivas, pero cambiando un parámetro en el modelo (concretamente maxdepth, que es la profundidad máxima del árbol) este puede llegar a mejorar los resultados obtenidos:

```
tctrl <- caret::trainControl(method = "repeatedcv",
                             number=10, repeats = 3)
tgrid <- data.frame(maxdepth = seq(2,10,1))

model_tree2<- train(Survived ~ Sex+FamilySize+Child+Age+Pclass,
                    data=data_train,
                    trControl=tctrl,
                    tuneGrid=tgrid,
```

```
                    method="rpart2")
```




```
## Call:
## (function (formula, data, weights, subset, na.action = na.rpart,
##     method, model = FALSE, x = FALSE, y = TRUE, parms, control,
##     cost, ...)
## {
##     Call <- match.call()
##     if (is.data.frame(model)) {
##         m <- model
##         model <- FALSE
##     }
##     else {
##         indx <- match(c("formula", "data", "weights", "subset"),
##             names(Call), nomatch = 0)
##         if (indx[1] == 0)
##             stop("a 'formula' argument is required")
##         temp <- Call[c(1, indx)]
##         temp$na.action <- na.action
##         temp[[1]] <- quote(stats::model.frame)
##         m <- eval.parent(temp)
##     }
##     Terms <- attr(m, "terms")
##     if (any(attr(Terms, "order") > 1))
##         stop("Trees cannot handle interaction terms")
##     Y <- model.response(m)
```

```
##     wt <- model.weights(m)
##     if (any(wt < 0))
##         stop("negative weights not allowed")
##     if (!length(wt))
##         wt <- rep(1, nrow(m))
##     offset <- model.offset(m)
##     X <- rpart.matrix(m)
##     nobs <- nrow(X)
##     nvar <- ncol(X)
##     if (missing(method)) {
##         method <- if (is.factor(Y) || is.character(Y))
##             "class"
##         else if (inherits(Y, "Surv"))
##             "exp"
##         else if (is.matrix(Y))
##             "poisson"
##         else "anova"
##     }
##     if (is.list(method)) {
##         mlist <- method
##         method <- "user"
##         init <- if (missing(parms))
##             mlist$init(Y, offset, wt = wt)
##         else mlist$init(Y, offset, parms, wt)
##         keep <- rpartcallback(mlist, nobs, init)
##         method.int <- 4
##         parms <- init$parms
##     }
##     else {
##         method.int <- pmatch(method, c("anova", "poisson", "class",
##             "exp"))
##         if (is.na(method.int))
##             stop("Invalid method")
##         method <- c("anova", "poisson", "class", "exp")[method.int]
##         if (method.int == 4)
##             method.int <- 2
##         init <- if (missing(parms))
##             get(paste("rpart", method, sep = "."), envir = environment())(Y,
##                 offset, , wt)
##         else get(paste("rpart", method, sep = "."), envir = environment())(Y,
##             offset, parms, wt)
##         ns <- asNamespace("rpart")
##         if (!is.null(init$print))
##             environment(init$print) <- ns
##         if (!is.null(init$summary))
##             environment(init$summary) <- ns
##         if (!is.null(init$text))
##             environment(init$text) <- ns
##     }
##     Y <- init$y
##     xlevels <- .getXlevels(Terms, m)
##     cats <- rep(0, ncol(X))
##     if (!is.null(xlevels))
##         cats[match(names(xlevels), colnames(X))] <- unlist(lapply(xlevels,
```

```
##             length))
##     extraArgs <- list(...)
##     if (length(extraArgs)) {
##         controlargs <- names(formals(rpart.control))
##         indx <- match(names(extraArgs), controlargs, nomatch = 0)
##         if (any(indx == 0))
##             stop(gettextf("Argument %s not matched", names(extraArgs)[indx ==
##                 0]), domain = NA)
##     }
##     controls <- rpart.control(...)
##     if (!missing(control))
##         controls[names(control)] <- control
##     xval <- controls$xval
##     if (is.null(xval) || (length(xval) == 1 && xval == 0) ||
##         method == "user") {
##         xgroups <- 0
##         xval <- 0
##     }
##     else if (length(xval) == 1) {
##         xgroups <- sample(rep(1:xval, length = nobs), nobs, replace = FALSE)
##     }
##     else if (length(xval) == nobs) {
##         xgroups <- xval
##         xval <- length(unique(xgroups))
##     }
##     else {
##         if (!is.null(attr(m, "na.action"))) {
##             temp <- as.integer(attr(m, "na.action"))
##             xval <- xval[-temp]
##             if (length(xval) == nobs) {
##                 xgroups <- xval
##                 xval <- length(unique(xgroups))
##             }
##             else stop("Wrong length for 'xval'")
##         }
##         else stop("Wrong length for 'xval'")
##     }
##     if (missing(cost))
##         cost <- rep(1, nvar)
##     else {
##         if (length(cost) != nvar)
##             stop("Cost vector is the wrong length")
##         if (any(cost <= 0))
##             stop("Cost vector must be positive")
##     }
##     tfun <- function(x) if (is.matrix(x))
##         rep(is.ordered(x), ncol(x))
##     else is.ordered(x)
##     labs <- sub("^`(.*)`$", "\\1", attr(Terms, "term.labels"))
##     isord <- unlist(lapply(m[labs], tfun))
##     storage.mode(X) <- "double"
##     storage.mode(wt) <- "double"
##     temp <- as.double(unlist(init$parms))
##     if (!length(temp))
```

```
##         temp <- 0
##     rpfit <- .Call(C_rpart, ncat = as.integer(cats * !isord),
##         method = as.integer(method.int), as.double(unlist(controls)),
##         temp, as.integer(xval), as.integer(xgroups), as.double(t(init$y)),
##         X, wt, as.integer(init$numy), as.double(cost))
##     nsplit <- nrow(rpfit$isplit)
##     ncat <- if (!is.null(rpfit$csplit))
##         nrow(rpfit$csplit)
##     else 0
##     if (nsplit == 0)
##         xval <- 0
##     numcp <- ncol(rpfit$cptable)
##     temp <- if (nrow(rpfit$cptable) == 3)
##         c("CP", "nsplit", "rel error")
##     else c("CP", "nsplit", "rel error", "xerror", "xstd")
##     dimnames(rpfit$cptable) <- list(temp, 1:numcp)
##     tname <- c("<leaf>", colnames(X))
##     splits <- matrix(c(rpfit$isplit[, 2:3], rpfit$dsplit), ncol = 5,
##         dimnames = list(tname[rpfit$isplit[, 1] + 1], c("count",
##             "ncat", "improve", "index", "adj")))
##     index <- rpfit$inode[, 2]
##     nadd <- sum(isord[rpfit$isplit[, 1]])
##     if (nadd > 0) {
##         newc <- matrix(0, nadd, max(cats))
##         cvar <- rpfit$isplit[, 1]
##         indx <- isord[cvar]
##         cdir <- splits[indx, 2]
##         ccut <- floor(splits[indx, 4])
##         splits[indx, 2] <- cats[cvar[indx]]
##         splits[indx, 4] <- ncat + 1:nadd
##         for (i in 1:nadd) {
##             newc[i, 1:(cats[(cvar[indx])[i]])] <- -as.integer(cdir[i])
##             newc[i, 1:ccut[i]] <- as.integer(cdir[i])
##         }
##         catmat <- if (ncat == 0)
##             newc
##         else {
##             cs <- rpfit$csplit
##             ncs <- ncol(cs)
##             ncc <- ncol(newc)
##             if (ncs < ncc)
##                 cs <- cbind(cs, matrix(0, nrow(cs), ncc - ncs))
##             rbind(cs, newc)
##         }
##         ncat <- ncat + nadd
##     }
##     else catmat <- rpfit$csplit
##     if (nsplit == 0) {
##         frame <- data.frame(row.names = 1, var = "<leaf>", n = rpfit$inode[,
##             5], wt = rpfit$dnode[, 3], dev = rpfit$dnode[, 1],
##             yval = rpfit$dnode[, 4], complexity = rpfit$dnode[,
##                 2], ncompete = 0, nsurrogate = 0)
##     }
##     else {
```

```
##         temp <- ifelse(index == 0, 1, index)
##         svar <- ifelse(index == 0, 0, rpfit$isplit[temp, 1])
##         frame <- data.frame(row.names = rpfit$inode[, 1], var = tname[svar +
##             1], n = rpfit$inode[, 5], wt = rpfit$dnode[, 3],
##             dev = rpfit$dnode[, 1], yval = rpfit$dnode[, 4],
##             complexity = rpfit$dnode[, 2], ncompete = pmax(0,
##                 rpfit$inode[, 3] - 1), nsurrogate = rpfit$inode[,
##                 4])
##     }
##     if (method.int == 3) {
##         numclass <- init$numresp - 2
##         nodeprob <- rpfit$dnode[, numclass + 5]/sum(wt)
##         temp <- pmax(1, init$counts)
##         temp <- rpfit$dnode[, 4 + (1:numclass)] %*% diag(init$parms$prior/temp)
##         yprob <- temp/rowSums(temp)
##         yval2 <- matrix(rpfit$dnode[, 4 + (0:numclass)], ncol = numclass +
##             1)
##         frame$yval2 <- cbind(yval2, yprob, nodeprob)
##     }
##     else if (init$numresp > 1)
##         frame$yval2 <- rpfit$dnode[, -(1:3), drop = FALSE]
##     if (is.null(init$summary))
##         stop("Initialization routine is missing the 'summary' function")
##     functions <- if (is.null(init$print))
##         list(summary = init$summary)
##     else list(summary = init$summary, print = init$print)
##     if (!is.null(init$text))
##         functions <- c(functions, list(text = init$text))
##     if (method == "user")
##         functions <- c(functions, mlist)
##     where <- rpfit$which
##     names(where) <- row.names(m)
##     ans <- list(frame = frame, where = where, call = Call, terms = Terms,
##         cptable = t(rpfit$cptable), method = method, parms = init$parms,
##         control = controls, functions = functions, numresp = init$numresp)
##     if (nsplit)
##         ans$splits = splits
##     if (ncat > 0)
##         ans$csplit <- catmat + 2
##     if (nsplit)
##         ans$variable.importance <- importance(ans)
##     if (model) {
##         ans$model <- m
##         if (missing(y))
##             y <- FALSE
##     }
##     if (y)
##         ans$y <- Y
##     if (x) {
##         ans$x <- X
##         ans$wt <- wt
##     }
##     ans$ordered <- isord
##     if (!is.null(attr(m, "na.action")))
```

```
##         ans$na.action <- attr(m, "na.action")
##     if (!is.null(xlevels))
##         attr(ans, "xlevels") <- xlevels
##     if (method == "class")
##         attr(ans, "ylevels") <- init$ylevels
##     class(ans) <- "rpart"
##     ans
## })(formula = .outcome ~ ., data = list(c(0, 0, 0, 1, 0, 1, 1,
## 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1,
## 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1,
## 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1,
## 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0,
## 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0,
## 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0,
## 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1,
## 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1,
## 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1,
## 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1,
## 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0,
## 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0,
## 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0,
## 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1,
## 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1,
## 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1,
## 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0,
## 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1,
## 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1,
## 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1,
## 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1,
## 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1,
## 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0,
## 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,
## 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1,
## 1, 1, 0, 1, 0, 1, 0, 1, 1, 0), c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
## 2, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
## 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
## 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 6, 3, 3, 5,
## 5, 6, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 2, 2, 2, 2, 3, 3, 3, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
## 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
## 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 5, 3, 3, 4, 4, 4, 4, 6, 1, 1,
```

```
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 5, 6, 2,
## 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
## 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3,
## 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 5, 5, 6, 6, 7, 7,
## 8, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 5, 5, 4, 5, 5, 5, 5, 5, 6,
## 6, 6, 6, 6, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7, 7, 7, 7, 8, 8, 8, 8,
## 11, 11, 11, 11, 11, 11, 11), c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0,
## 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0,
## 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0,
## 0, 0, 0, 0, 0, 1, 0), c(30, 33, 16, 47, 22, 35, 33, 19, 21, 40,
## 40, 54, 45.5, 45, 18, 28, 29, 38, 30, 45, 29, 62, 62, 38, 64,
## 58, 42, 31, 47, 46, 50, 36, 56, 45, 52, 55, 28, 56, 34, 31, 60,
## 11, 71, 27, 62, 24, 42, 41, 32, 36, 24, 35, 30, 54, 40, 32, 48,
## 65, 24, 28, 56, 26, 32, 21, 24, 49, 24, 48, 61, 47, 44, 65, 58,
```

```
## 36, 35, 42, 35, 38, 27, 18, 38, 65, 58, 54, 16, 21, 22, 30, 36,
## 44, 15, 51, 24, 43, 50, 31, 36, 22, 58, 17, 27, 4, 19, 48, 35,
## 35, 31, 49, 33, 31, 27, 54, 50, 37, 49, 54, 35, 35, 45, 71, 35,
## 50, 46, 33, 18, 50, 25, 17, 42, 48, 25, 38, 27, 19, 60, 36, 38,
## 49, 52, 48, 18, 37, 70, 45, 49, 40, 60, 39, 47, 36, 14, 2, 36,
## 64, 50, 53, 18, 21, 24, 50, 34, 52, 18, 40, 40, 30, 28, 27, 27,
## 55, 30, 42, 27, 30, 42, 25, 30, 47, 34, 30, 24, 70, 28, 16, 36,
## 22, 30, 35, 34, 57, 27, 50, 19, 28, 19, 27, 26, 32, 48, 66, 21,
## 25, 23, 59, 45, 23, 28, 36, 54, 28, 31, 23, 60, 24, 24, 19, 39,
## 36, 31, 27, 32, 50, 34, 18, 36, 62, 38, 32.5, 39, 16, 19, 28,
## 21, 46, 54, 51, 29, 35, 24, 17, 28, 35, 25, 34, 28, 50, 7, 36.5,
## 18, 8, 24, 42, 27, 25, 29, 34, 32.5, 54, 29, 28, 29, 34, 27,
## 29, 19, 44, 21, 24, 36, 44, 30, 2, 45, 3, 43, 34, 40, 8, 3, 25,
## 31, 31, 31, 22, 2, 4, 24, 3, 36, 22, 48, 25, 33, 5, 54, 32, 21,
## 24, 1, 23, 4, 24, 36, 28, 24, 28, 17, 51, 17, 27, 24, 36, 32,
## 30, 19, 28, 21, 34, 47, 38, 30, 26, 18, 45.5, 18, 30, 20, 26,
## 30.5, 47, 51, 28, 29, 26, 26, 24, 22, 30.5, 50, 21, 20, 28, 26,
## 30, 17, 65, 41, 31, 20, 32, 28, 20, 21, 35, 16, 29, 42, 14, 23,
## 21, 32, 28, 47, 26, 13, 40, 31, 22, 34, 17, 24, 21, 29, 15, 18,
## 38, 17, 19, 21, 20, 18, 49, 56, 18, 43, 27, 28, 18, 25, 22, 22,
## 45.5, 26, 25, 55, 20, 17, 21, 61, 32, 29, 35, 19, 33, 38, 17,
## 28, 35, 32, 20, 22, 24, 15, 25, 35, 17, 11, 21, 17, 19, 21, 20.5,
## 33, 26, 27, 61, 22, 35, 5, 30, 47, 22, 25, 26, 45, 19, 35, 22,
## 33, 24, 23.5, 30, 45, 28.5, 51, 29, 27, 32, 16, 19, 44, 32, 18,
## 59, 26, 4, 22, 21, 32, 27, 20, 35, 45, 24, 54, 18, 19, 28, 20,
## 30, 25, 28, 20, 36, 48, 31, 16, 26, 25, 39, 16, 21, 13, 21, 20,
## 0.42, 19, 37, 43, 18, 25, 21, 51, 18, 22, 1, 29, 32, 36, 35,
## 33, 29, 20, 21, 32, 35, 32, 21, 22, 40.5, 30.5, 23, 22, 23, 47,
## 58, 22, 30, 48, 42, 21, 28.5, 42, 20, 26, 19, 20.5, 23.5, 70.5,
## 26, 29, 20, 36, 38, 55.5, 41, 44, 27, 18, 42, 2, 18, 41, 10,
## 27, 24, 40.5, 58, 31, 29, 19, 1, 24, 29, 41, 19, 24, 12, 44,
## 38, 36, 30, 25, 18, 25, 32, 4, 29, 47, 19, 25, 21, 22, 27, 21,
## 24, 19, 31, 26, 26, 20, 15, 24, 35, 20, 40, 23, 36, 17, 15, 33,
## 4, 4, 16, 18, 34, 18, 35, 20, 30, 30, 36, 9, 9, 40, 32, 29, 35,
## 26, 17, 1, 18, 26, 48, 16, 45, 40, 39, 38, 43, 18, 18, 28, 27,
## 16, 41, 24, 37, 32, 5, 0.75, 21, 9, 33, 15, 2, 51, 8, 3, 9, 10,
## 2, 7, 7, 2, 14, 16, 8, 1, 3, 5, 2, 17, 4, 11, 9, 6, 1, 14, 11,
## 16, 41, 9, 23, 22, 28, 2, 18), c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0), c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1), c(2, 2, 2, 1, 2, 2, 1, 2, 1, 1, 2, 1, 1, 1, 1, 1,
## 1, 2, 2, 1, 2, 1, 2, 2, 1, 1, 2, 1, 1, 1, 1, 1, 2, 1, 2, 1, 2,
## 1, 2, 1, 1, 1, 1, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 1,
## 1, 2, 1, 2, 2, 2, 2, 2, 2, 2, 1, 1, 2, 1, 2, 2, 2, 2, 2, 1, 2,
## 1, 1, 1, 2, 1, 2, 1, 2, 2, 2, 2, 2, 1, 1, 2, 2, 2, 2, 2, 1, 2,
## 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 2, 2, 2, 2, 1, 2, 2,
## 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 1, 2,
```

```
## 1, 2, 1, 2, 2, 2, 2, 2, 2, 1, 2, 1, 2, 2, 2, 2, 2, 2, 1, 1, 1,
## 2, 2, 2, 2, 1, 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1,
## 2, 1, 1, 2, 2, 1, 1, 2, 2, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 2,
## 1, 1, 1, 1, 1, 2, 1, 1, 2, 1, 2, 1, 1, 1, 2, 2, 1, 2, 1, 2, 2,
## 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 2, 2, 1, 2, 2, 1, 2, 2,
## 1, 2, 2, 2, 1, 2, 1, 1, 1, 1, 1, 2, 2, 2, 1, 1, 1, 2, 1, 1, 2,
## 2, 1, 1, 2, 2, 2, 1, 2, 2, 2, 2, 2, 1, 2, 1, 2, 2, 2, 2, 2, 1,
## 2, 2, 1, 2, 2, 2, 1, 1, 2, 2, 1, 2, 2, 1, 1, 1, 1, 2, 1, 1, 2,
## 2, 1, 2, 1, 2, 1, 1, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 2, 2, 1, 1,
## 1, 2, 1, 1, 1, 2, 1, 2, 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1,
## 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 2, 1,
## 1, 2, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 2, 2, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 2, 1, 2, 2, 1, 1, 1, 1, 2, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1,
## 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 2, 1, 2, 1, 1, 1, 1, 1, 2, 1,
## 1, 1, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 2,
## 1, 1, 1, 2, 2, 1, 1, 1, 2, 2, 2, 2, 1, 1, 2, 2, 2, 1, 2, 2, 1,
## 1, 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 2, 1,
## 1, 2, 1, 2, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 2, 2, 1, 2, 1, 1, 2,
## 1, 1, 1, 2, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1,
## 1, 1, 1, 2, 2, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1, 1, 1, 2, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
## 1)), control = list(20, 7, 0.01, 4, 5, 2, 0, 3, 0))
##   n= 668
##
##           CP nsplit rel error
## 1 0.47265625      0 1.0000000
## 2 0.03125000      1 0.5273438
## 3 0.01757812      3 0.4648438
## 4 0.01000000      5 0.4296875
##
## Variable importance
##    Sexmale    Pclass3 FamilySize        Age    Pclass2     Child1
##         55         16         13          7          6          3
##
## Node number 1: 668 observations,    complexity param=0.4726562
##   predicted class=0  expected loss=0.3832335  P(node) =1
##     class counts:   412   256
##    probabilities: 0.617 0.383
##   left son=2 (427 obs) right son=3 (241 obs)
##   Primary splits:
##       Sexmale    < 0.5  to the right, improve=102.006600, (0 missing)
##       Pclass3    < 0.5  to the right, improve= 35.590460, (0 missing)
##       FamilySize < 1.5  to the left,  improve= 12.830360, (0 missing)
##       Age        < 5.5  to the right, improve=  4.986881, (0 missing)
##       Child1     < 0.5  to the left,  improve=  4.028764, (0 missing)
##   Surrogate splits:
##       FamilySize < 1.5  to the left,  agree=0.659, adj=0.054, (0 split)
##       Age        < 15.5 to the right, agree=0.644, adj=0.012, (0 split)
##
## Node number 2: 427 observations,    complexity param=0.01757812
```

```
##   predicted class=0  expected loss=0.175644  P(node) =0.6392216
##     class counts:   352    75
##    probabilities: 0.824 0.176
##   left son=4 (404 obs) right son=5 (23 obs)
##   Primary splits:
##       Age        < 9.5  to the right, improve=4.4523630, (0 missing)
##       Child1     < 0.5  to the left,  improve=3.9900130, (0 missing)
##       Pclass3    < 0.5  to the right, improve=3.4730370, (0 missing)
##       FamilySize < 1.5  to the left,  improve=2.3760430, (0 missing)
##       Pclass2    < 0.5  to the right, improve=0.6269878, (0 missing)
##   Surrogate splits:
##       Child1 < 0.5  to the left,  agree=0.995, adj=0.913, (0 split)
##
## Node number 3: 241 observations,    complexity param=0.03125
##   predicted class=1  expected loss=0.2489627  P(node) =0.3607784
##     class counts:    60   181
##    probabilities: 0.249 0.751
##   left son=6 (108 obs) right son=7 (133 obs)
##   Primary splits:
##       Pclass3    < 0.5  to the right, improve=24.66583000, (0 missing)
##       FamilySize < 4.5  to the right, improve=12.57550000, (0 missing)
##       Pclass2    < 0.5  to the left,  improve= 5.30974800, (0 missing)
##       Age        < 18.5 to the left,  improve= 2.80259000, (0 missing)
##       Child1     < 0.5  to the right, improve= 0.06214367, (0 missing)
##   Surrogate splits:
##       Pclass2    < 0.5  to the left,  agree=0.697, adj=0.324, (0 split)
##       Age        < 23.5 to the left,  agree=0.676, adj=0.278, (0 split)
##       FamilySize < 4.5  to the right, agree=0.622, adj=0.157, (0 split)
##       Child1     < 0.5  to the right, agree=0.573, adj=0.046, (0 split)
##
## Node number 4: 404 observations
##   predicted class=0  expected loss=0.1584158  P(node) =0.6047904
##     class counts:   340    64
##    probabilities: 0.842 0.158
##
## Node number 5: 23 observations,    complexity param=0.01757812
##   predicted class=0  expected loss=0.4782609  P(node) =0.03443114
##     class counts:    12    11
##    probabilities: 0.522 0.478
##   left son=10 (12 obs) right son=11 (11 obs)
##   Primary splits:
##       FamilySize < 4.5  to the right, improve=7.8267460, (0 missing)
##       Age        < 2.5  to the left,  improve=0.2167224, (0 missing)
##   Surrogate splits:
##       Pclass3 < 0.5  to the right, agree=0.783, adj=0.545, (0 split)
##       Pclass2 < 0.5  to the left,  agree=0.739, adj=0.455, (0 split)
##       Age     < 1.5  to the right, agree=0.565, adj=0.091, (0 split)
##
## Node number 6: 108 observations,    complexity param=0.03125
##   predicted class=0  expected loss=0.5  P(node) =0.1616766
##     class counts:    54    54
##    probabilities: 0.500 0.500
##   left son=12 (22 obs) right son=13 (86 obs)
##   Primary splits:
```

```
##       FamilySize < 4.5  to the right, improve=7.3065540, (0 missing)
##       Age        < 39   to the right, improve=4.9090910, (0 missing)
##       Child1     < 0.5  to the left,  improve=0.3935223, (0 missing)
##   Surrogate splits:
##       Age < 40.5 to the right, agree=0.815, adj=0.091, (0 split)
##
## Node number 7: 133 observations
##   predicted class=1  expected loss=0.04511278  P(node) =0.1991018
##     class counts:     6   127
##    probabilities: 0.045 0.955
##
## Node number 10: 12 observations
##   predicted class=0  expected loss=0.08333333  P(node) =0.01796407
##     class counts:    11     1
##    probabilities: 0.917 0.083
##
## Node number 11: 11 observations
##   predicted class=1  expected loss=0.09090909  P(node) =0.01646707
##     class counts:     1    10
##    probabilities: 0.091 0.909
##
## Node number 12: 22 observations
##   predicted class=0  expected loss=0.1363636  P(node) =0.03293413
##     class counts:    19     3
##    probabilities: 0.864 0.136
##
## Node number 13: 86 observations
##   predicted class=1  expected loss=0.4069767  P(node) =0.1287425
##     class counts:    35    51
##    probabilities: 0.407 0.593
```

Realizamos la predicción con el modelo construido con rpart y el dataset test

```
##    predict_tree2
##       0   1
##   0 119  18
##   1  25  61
```

```
## [1] "El % de registros correctamente clasificados es: 80.7175 %"
```

```
## [1] "El mejor parámetro para el árbol de clasificación ha sido: maxdepth=3"
```

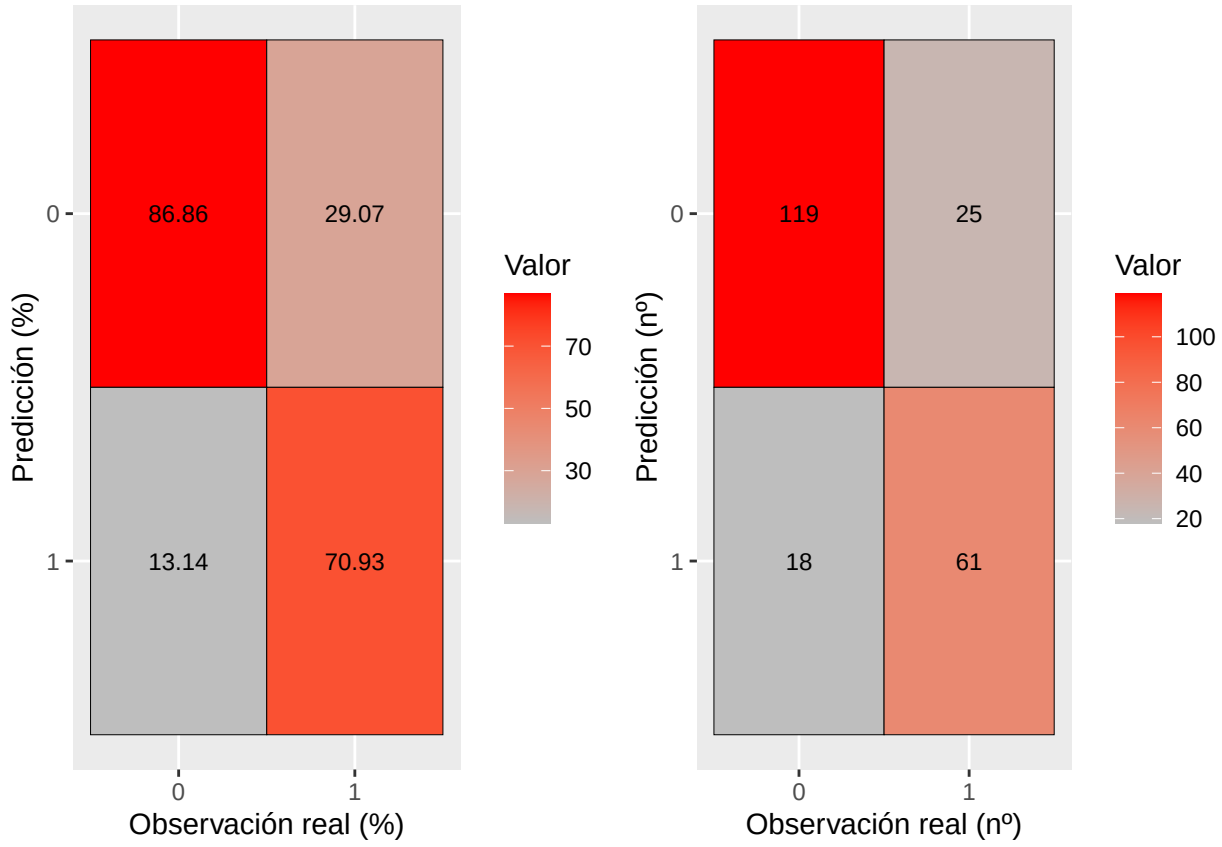
Matriz de confusión

```
csstab_tree2 <- CrossTable(data_test$Survived, predict_tree2,
                           prop.chisq  = FALSE, prop.c = FALSE, prop.r =FALSE,
                           dnn = c('Reality', 'Prediction'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
```

```
## Total Observations in Table:  223
##
##
##                | Prediction
##      Reality |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |       119 |        18 |       137 |
##              |     0.534 |     0.081 |           |
## -------------|-----------|-----------|-----------|
##            1 |        25 |        61 |        86 |
##              |     0.112 |     0.274 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       144 |        79 |       223 |
## -------------|-----------|-----------|-----------|
##
##
```

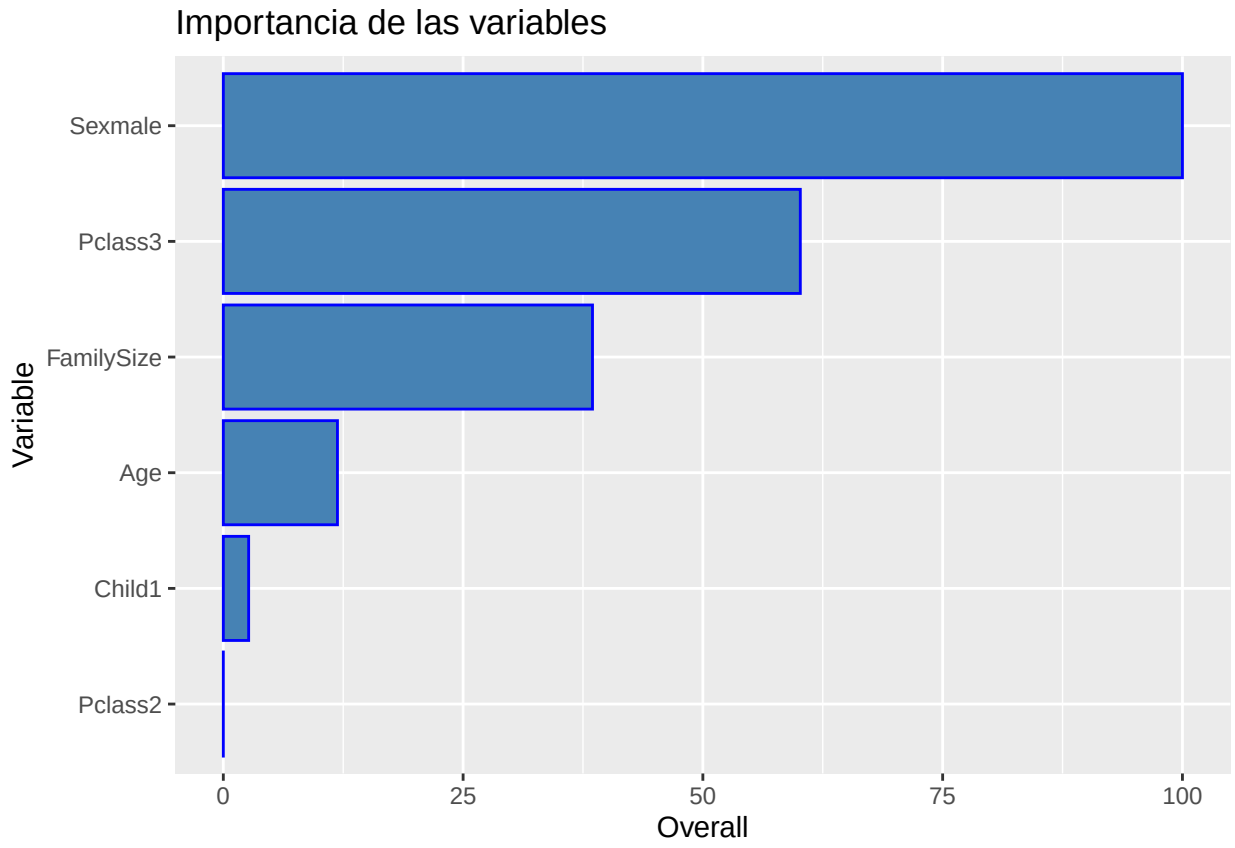
```
plot_matriz_confusion(csstab_tree2)
```



Importancia de las variables

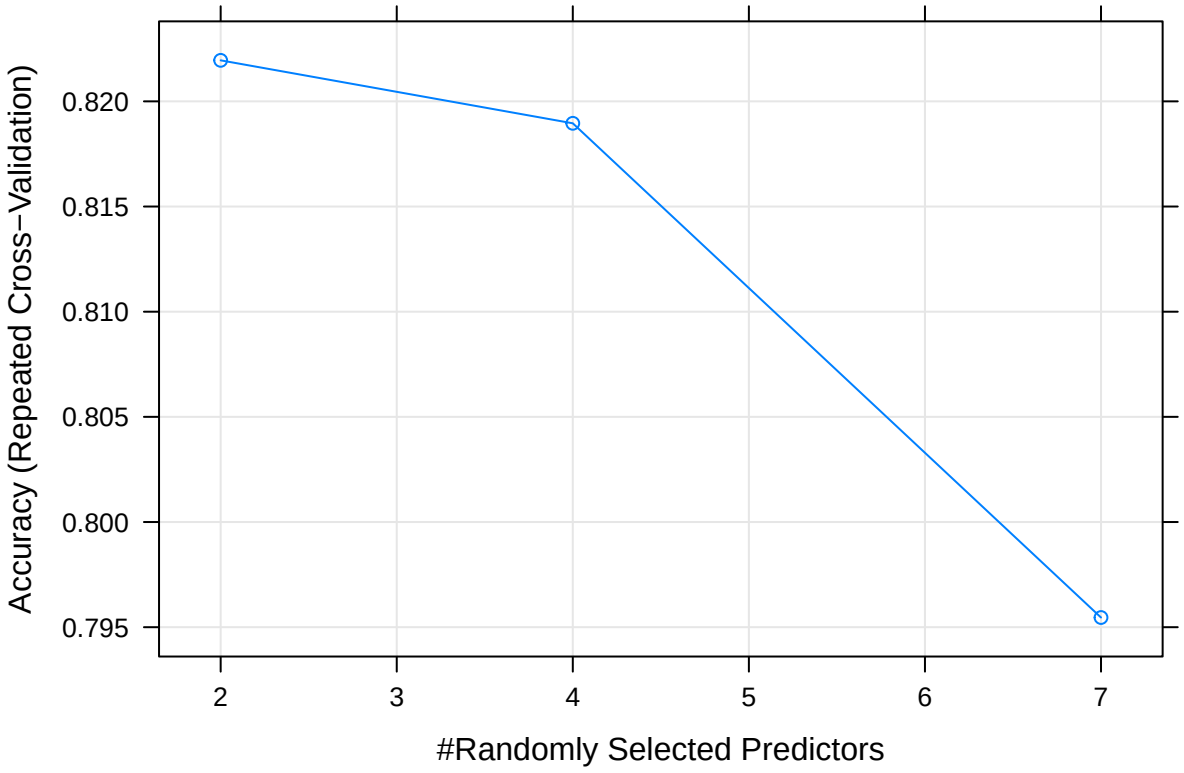
```
datosVarImp <- varImp(model_tree2)$importance %>%
  as.data.frame() %>%
  rownames_to_column() %>%
  arrange(Overall)
datosVarImp$Variable <- reorder(datosVarImp$rowname, datosVarImp$Overall)
ggplot(datosVarImp)+
```

```
  geom_col(aes(x = Variable, y = Overall), fill="steelblue", color="blue")+
  coord_flip() + ggtitle("Importancia de las variables")
```



4.3.2.3 Random Forest Utilizaremos ahora Random Forest, que consiste en una combinación de árboles predictores. Cada árbol utiliza muestras con reemplazo del conjunto de datos que se le pasa al modelo. Cada árbol utiliza valores diferentes de variables, para de esa forma dar también opciones a algunas variables que podrían quedar eclipsadas por otras con más relevancia. Y de ahí que luego se pueda obtener la importancia relativa de cada variable (el clasificador altera la variable y mide el impacto). El uso de esta técnica nos va a permitir tener, respecto a los árboles de clasificación, un sistema más estable.

```
tctrl <- caret::trainControl(method = "repeatedcv",
                             number=10, repeats = 3)
model_rf <- caret::train(Survived ~ Sex+Parch+SibSp+Child+Age+Pclass,
                         data = data_train,
                         method = "rf",
                         trControl = tctrl,
                         verbose = FALSE)
plot(model_rf)
```



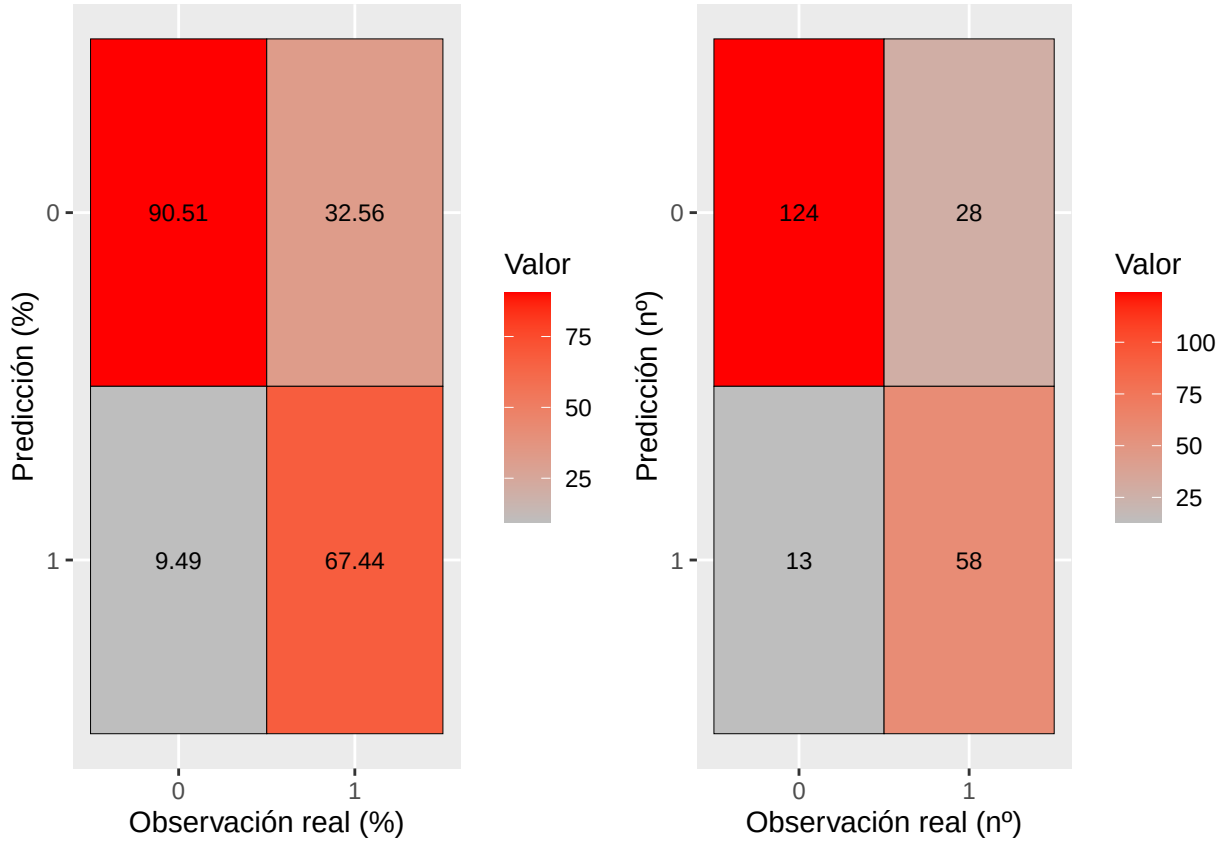
Matriz de confusión

```
##    predict_rf
##       0   1
##   0 124  13
##   1  28  58
```

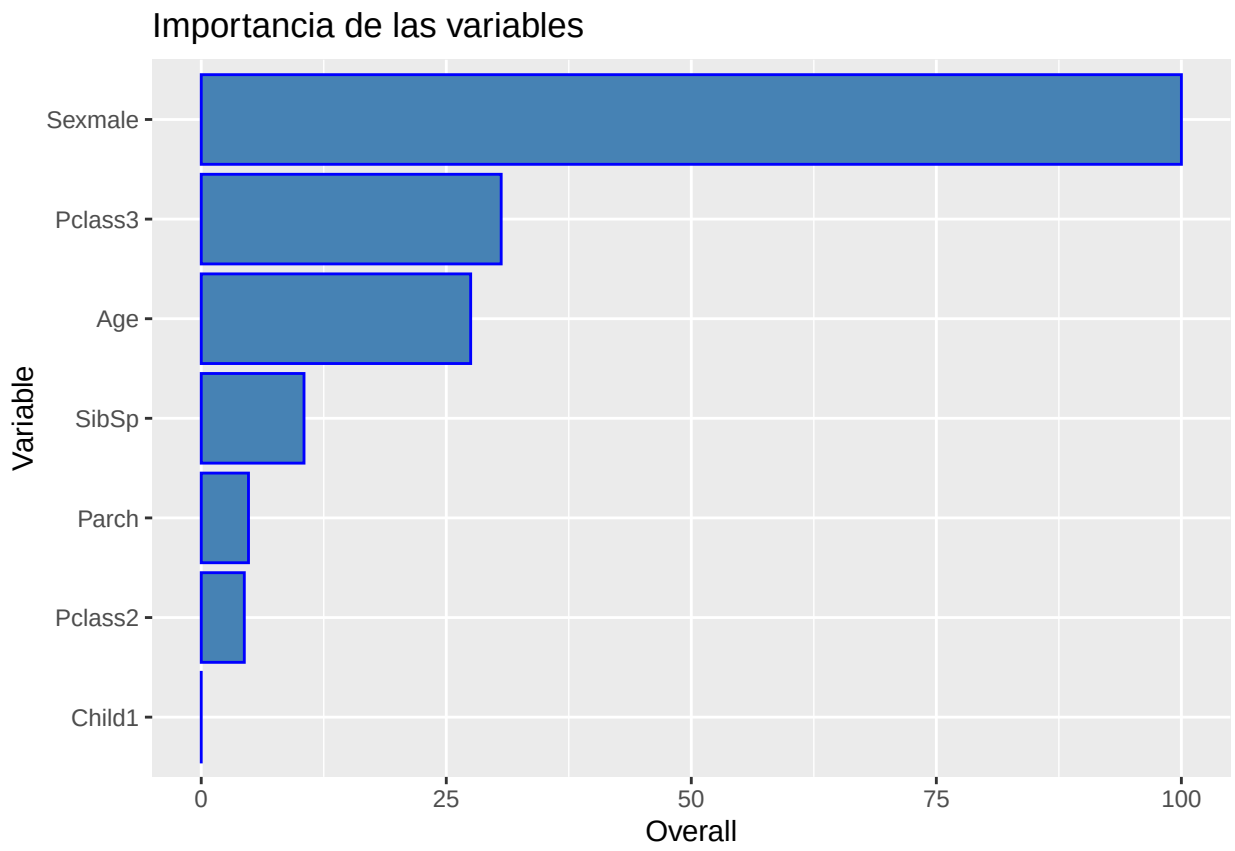
```
## [1] "El % de registros correctamente clasificados es: 81.6143 %"
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  223
##
##
##              | Prediction
##      Reality |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |       124 |        13 |       137 |
##              |     0.556 |     0.058 |           |
## -------------|-----------|-----------|-----------|
##            1 |        28 |        58 |        86 |
```

```
##              |     0.126 |     0.260 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       152 |        71 |       223 |
## -------------|-----------|-----------|-----------|
##
##
```



Importancia de las variables



4.3.2.4 C5.0 C5.0 es el algoritmo sucesor de C4.5, publicado por John Ross Quinlan (1992), con el objetivo de crear árboles de clasificación. Entre sus características, destacan la capacidad para generar árboles de decisión simples, modelos basados en reglas, ensembles basados en boosting y asignación de distintos pesos a los errores. Este algoritmo ha resultado de una gran utilidad a la hora de crear modelos de clasificación y todas sus capacidades son accesibles mediante el paquete C50. Aunque comparte muchas características con los algoritmos de random forest y gradient boosting, cabe tener en cuenta algunas peculiaridades:

- La medida de pureza empleada para las divisiones del árbol es la entropía.
- El podado de los árboles se realiza por defecto, y el método empleado se conoce como pessimistic pruning.
- Los árboles se pueden convertir en modelos basados en reglas.
- Emplea un algoritmo de boosting más próximo a AdaBoost que a Gradient Boosting.
- Por defecto, el algoritmo de boosting se detiene si la incorporación de nuevos modelos no aporta un mínimo de mejora.
- Incorpora una estrategia para la selección de predictores (Winnowing) previo ajuste del modelo.
- Permite asignar diferente peso a cada tipo de error.

Creamos primero unas variables para simplificar el uso de los distintos modelos. - trainX corresponde al dataframe sin la variable Survived (para el conjunto de train) - trainy es el dataframe pero sólo con la variable Survived (para el conjunto de train)

- testX corresponde al dataframe sin la variable Survived (para el conjunto de test)
- testy es el dataframe pero sólo con la variable Survived (para el conjunto de test)

```
trainX <- data_train[, c(1:3,5:12)]
trainy <- data_train[, 4]
testX <- data_test[, c(1:3,5:12)]
testy <- data_test[, 4]
```

4.3.2.4.1 Importancia Atributos Definimos primero las métricas que vamos a usar en los árboles C5.0 a la hora de mostrar la importancia de los atributos:

- Usage: porcentaje de observaciones de entrenamiento que caen en todos los nodos generados tras una división en el que ha participado el predictor. Por ejemplo, en el caso de un árbol simple, el predictor empleado en la primera decisión recibe automáticamente una importancia del 100%, ya que todas las observaciones de entrenamiento caen en uno de los dos nodos hijos generados. Otros predictores puede que participen con frecuencia en divisiones, pero si en los nodos hijos solo caen unas pocas observaciones, su importancia puede ser próxima a cero. En el caso de boosting, se promedia la importancia de cada predictor en todos los árboles que forman el ensemble.
- Splits: porcentaje de divisiones en las que participa cada predictor. No tiene en cuenta la importancia de la división.

4.3.2.4.2 Primer modelo c5.0 Vamos a empezar realizando un primer modelo C5.0 con los valores por defecto y con las reglas:

```
bc_tree <- C5.0(trainX, trainy, rules=TRUE )
summary(bc_tree)
```

```
##
## Call:
## C5.0.default(x = trainX, y = trainy, rules = TRUE)
##
##
## C5.0 [Release 2.07 GPL Edition]      Mon Jan  4 06:01:40 2021
## -------------------------------
##
## Class specified by attribute `outcome'
##
## Read 668 cases (12 attributes) from undefined.data
##
## Rules:
##
## Rule 1: (23, lift 1.6)
##  Pclass = 3
##  FamilySize > 4
##  FamilySize <= 6
##  ->  class 0  [0.960]
##
## Rule 2: (15, lift 1.5)
##  Price <= 5.825
##  FamilySize > 4
##  ->  class 0  [0.941]
##
## Rule 3: (47/2, lift 1.5)
##  Age > 27
##  Fare <= 8.1125
##  Embarked = S
##  AgeInterval = 18-39
```

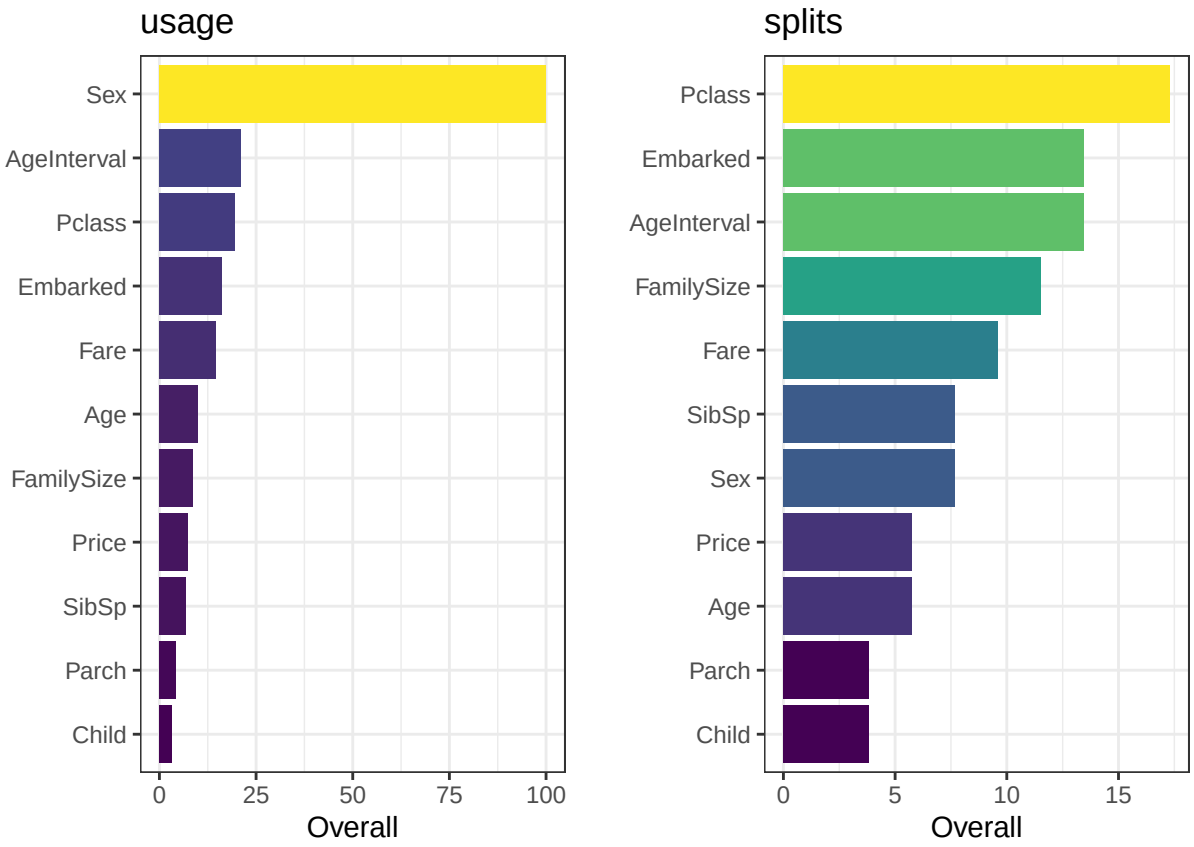
```
##  ->  class 0  [0.939]
##
## Rule 4: (13, lift 1.5)
##  Pclass = 3
##  Age <= 17
##  Embarked = S
##  FamilySize <= 4
##  AgeInterval = 13-17
##  ->  class 0  [0.933]
##
## Rule 5: (12, lift 1.5)
##  Pclass = 3
##  Sex = female
##  Embarked = S
##  Child = 0
##  AgeInterval in {0-12, 40-54}
##  ->  class 0  [0.929]
##
## Rule 6: (12, lift 1.5)
##  FamilySize > 7
##  ->  class 0  [0.929]
##
## Rule 7: (8, lift 1.5)
##  Pclass = 3
##  SibSp > 0
##  Embarked = S
##  FamilySize <= 4
##  AgeInterval = 13-17
##  ->  class 0  [0.900]
##
## Rule 8: (23/2, lift 1.4)
##  Pclass = 3
##  Parch <= 1
##  Fare > 8.1125
##  Fare <= 11.2417
##  AgeInterval = 18-39
##  ->  class 0  [0.880]
##
## Rule 9: (6, lift 1.4)
##  Parch > 0
##  Embarked = Q
##  ->  class 0  [0.875]
##
## Rule 10: (427/75, lift 1.3)
##  Sex = male
##  ->  class 0  [0.823]
##
## Rule 11: (16, lift 2.5)
##  Pclass = 1
##  SibSp > 0
##  Fare <= 91.0792
##  Embarked = C
##  ->  class 1  [0.944]
##
```

```
## Rule 12: (9, lift 2.4)
##  SibSp <= 2
##  Sex = male
##  Embarked = S
##  Child = 1
##  ->  class 1  [0.909]
##
## Rule 13: (34/3, lift 2.3)
##  Pclass = 1
##  Price > 12.65
##  Price <= 37.75834
##  AgeInterval = 18-39
##  ->  class 1  [0.889]
##
## Rule 14: (6, lift 2.3)
##  Pclass = 1
##  Age <= 43
##  AgeInterval = 40-54
##  ->  class 1  [0.875]
##
## Rule 15: (13/1, lift 2.3)
##  Pclass = 1
##  SibSp > 0
##  Fare <= 53.1
##  ->  class 1  [0.867]
##
## Rule 16: (241/60, lift 2.0)
##  Sex = female
##  ->  class 1  [0.749]
##
## Default class: 0
##
##
## Evaluation on training data (668 cases):
##
##          Rules
##    ----------------
##      No      Errors
##
##      16   66( 9.9%)   <<
##
##
##     (a)   (b)    <-classified as
##    ----  ----
##     387    25    (a): class 0
##      41   215    (b): class 1
##
##
##  Attribute usage:
##
##  100.00% Sex
##   20.96% AgeInterval
##   19.46% Pclass
##   16.17% Embarked
```

```
##   14.67% Fare
##    9.88% Age
##    8.68% FamilySize
##    7.34% Price
##    6.74% SibSp
##    4.34% Parch
##    3.14% Child
##
##
## Time: 0.0 secs
```

Aunque podríamos representar el árbol, nos han salido muchas reglas y es complicado de visualizar. Lo analizamos numéricamente.

Primero veamos visualmente la importancia de cada variable:

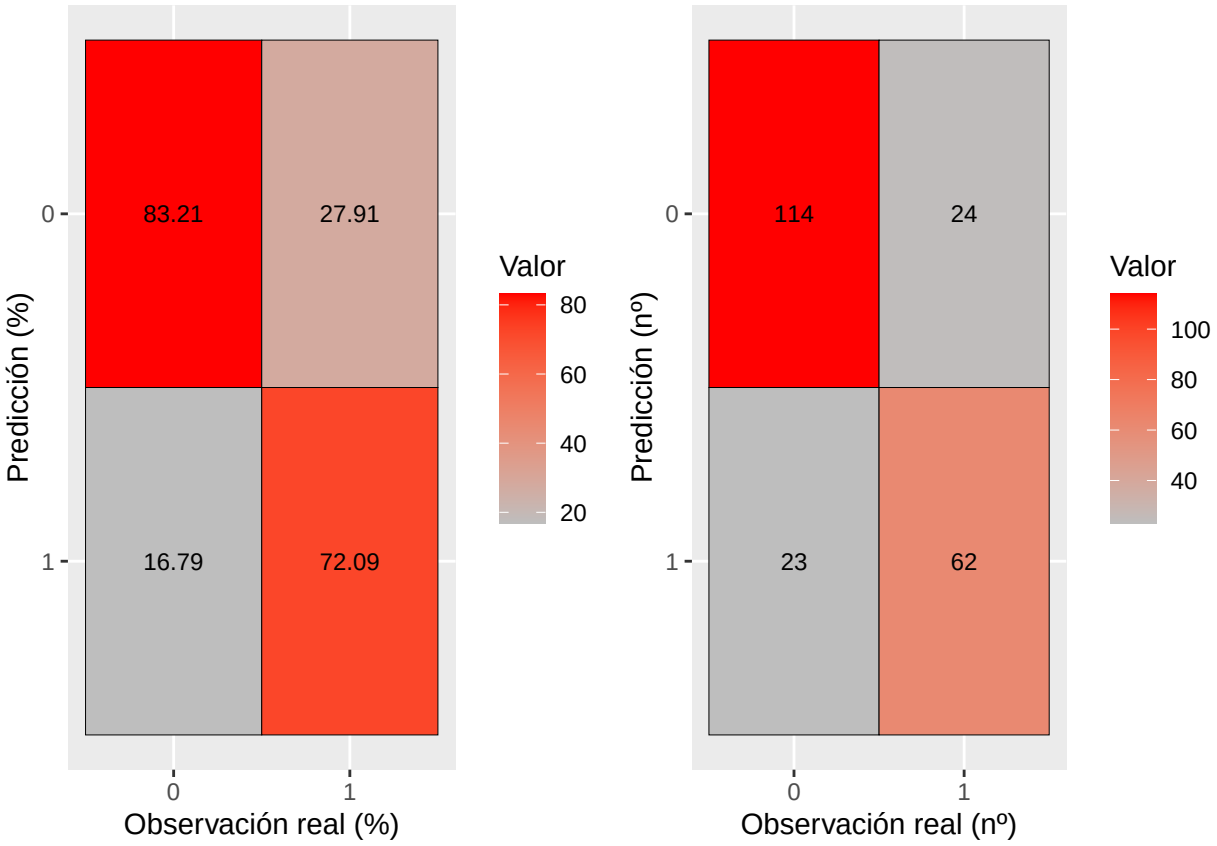


Está usando casi todas las variables, de ahí que nos salgan tantas reglas.

Vamos a ver la precisión del modelo

```
## [1] "La precisión del árbol C50 1 es: 78.9238 %"
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
```

```
##
##
## Total Observations in Table:  223
##
##
##               | Prediction
##      Reality |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |       114 |        23 |       137 |
##              |     0.826 |     0.271 |           |
##              |     0.511 |     0.103 |           |
## -------------|-----------|-----------|-----------|
##            1 |        24 |        62 |        86 |
##              |     0.174 |     0.729 |           |
##              |     0.108 |     0.278 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       138 |        85 |       223 |
##              |     0.619 |     0.381 |           |
## -------------|-----------|-----------|-----------|
##
##
```



4.3.2.4.3 Segundo modelo C5.0 con Winnowing Winnowing consiste en aplicar un algoritmo de clasificación (algoritmo Winnow) a los atributos para eliminar aquellos que sean de poca ayuda.

Es una selección de atributos que se realiza antes de la creación del modelo.

El conjunto de datos se divide por la mitad aleatoriamente y se crea un modelo inicial. Cada predictor se va quitando por turno y se analiza el efecto sobre la calidad del modelo (usando la otra mitad de la división aleatoria)

Se marcan los predictores si su eliminación no incrementa la tasa de error. El modelo final se calcula usando todos los conjuntos de datos usando sólo los predictores no marcados.

Se puede realizar winnowing en C5.0 simplemente usando un flag.

```
bc_winno_tree <- C5.0(
                      formula = Survived ~ .,
                      data    = data_train,
                      rules   = TRUE,
                      control = C5.0Control(seed = 123, winnow = TRUE)
                     )
summary(bc_winno_tree)
```

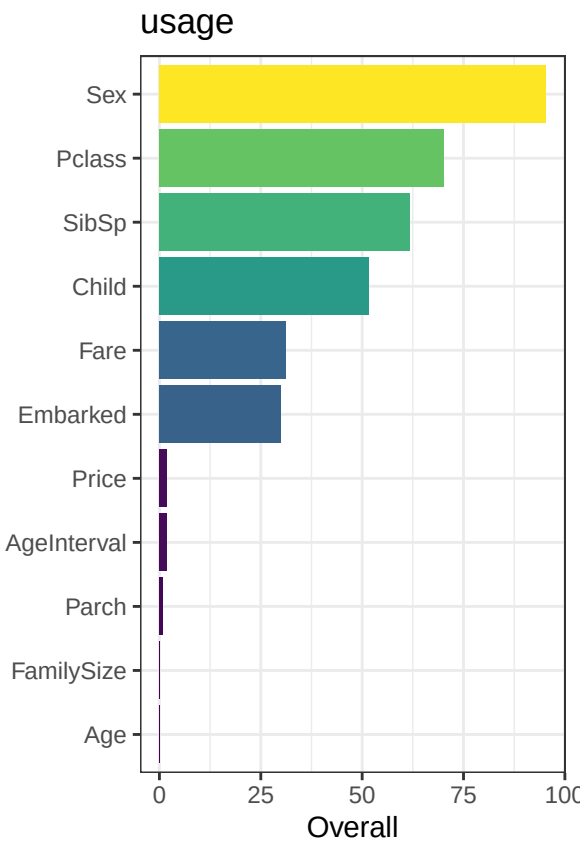
```
##
## Call:
## C5.0.formula(formula = Survived ~ ., data = data_train, rules = TRUE, control
##  = C5.0Control(seed = 123, winnow = TRUE))
##
##
## C5.0 [Release 2.07 GPL Edition]      Mon Jan  4 06:01:41 2021
## -------------------------------
##
## Class specified by attribute `outcome'
##
## Read 668 cases (12 attributes) from undefined.data
##
## 1 attribute winnowed
## Estimated importance of remaining attributes:
##
##      74%  Sex
##       8%  Pclass
##       6%  Price
##       5%  AgeInterval
##       3%  Embarked
##       2%  Fare
##      <1%  SibSp
##      <1%  Parch
##      <1%  FamilySize
##      <1%  Child
##
## Rules:
##
## Rule 1: (20/1, lift 1.5)
##  SibSp > 2
##  Sex = male
##  ->  class 0  [0.909]
##
## Rule 2: (321/33, lift 1.5)
##  Pclass in {2, 3}
##  Sex = male
##  Child = 0
##  ->  class 0  [0.895]
```

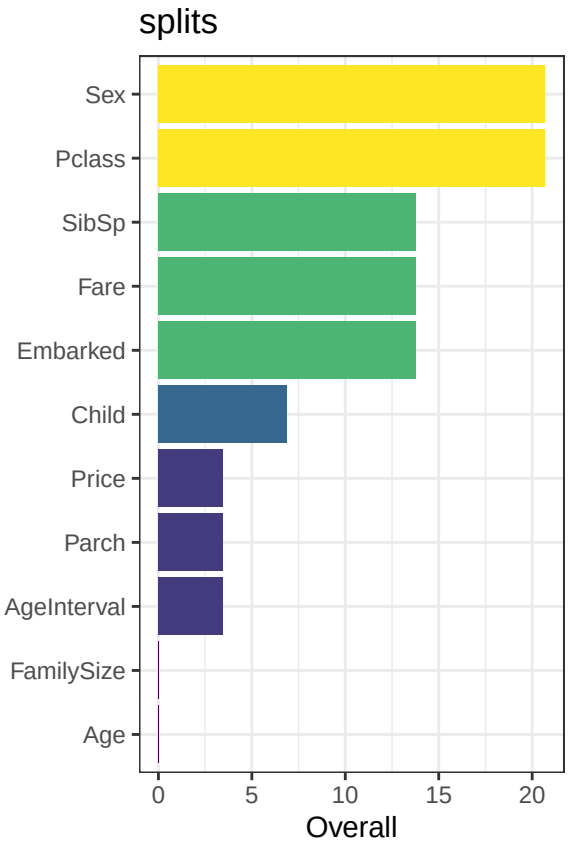
```
##
## Rule 3: (6, lift 1.4)
##  Parch > 0
##  Embarked = Q
##  ->  class 0  [0.875]
##
## Rule 4: (28/3, lift 1.4)
##  Pclass = 3
##  Sex = female
##  Fare > 17.4
##  Embarked = S
##  ->  class 0  [0.867]
##
## Rule 5: (167/23, lift 1.4)
##  Pclass = 3
##  Fare <= 10.5
##  Embarked = S
##  ->  class 0  [0.858]
##
## Rule 6: (314/45, lift 1.4)
##  SibSp <= 0
##  Sex = male
##  ->  class 0  [0.854]
##
## Rule 7: (24/2, lift 2.3)
##  Pclass in {2, 3}
##  SibSp <= 2
##  Child = 1
##  ->  class 1  [0.885]
##
## Rule 8: (13/1, lift 2.3)
##  Pclass = 1
##  Fare <= 37.0042
##  Price > 12.65
##  AgeInterval = 18-39
##  ->  class 1  [0.867]
##
## Rule 9: (3, lift 2.1)
##  Sex = female
##  Fare <= 7.75
##  Embarked = S
##  ->  class 1  [0.800]
##
## Rule 10: (56/11, lift 2.1)
##  Pclass = 1
##  SibSp > 0
##  ->  class 1  [0.793]
##
## Rule 11: (241/60, lift 2.0)
##  Sex = female
##  ->  class 1  [0.749]
##
## Default class: 0
##
```

```
##
## Evaluation on training data (668 cases):
##
##          Rules
##    ----------------
##      No      Errors
##
##      11   82(12.3%)   <<
##
##
##     (a)   (b)    <-classified as
##    ----  ----
##     383    29    (a): class 0
##      53   203    (b): class 1
##
##
##  Attribute usage:
##
##   95.36% Sex
##   70.06% Pclass
##   61.68% SibSp
##   51.65% Child
##   31.14% Fare
##   30.09% Embarked
##    1.95% Price
##    1.95% AgeInterval
##    0.90% Parch
##
##
## Time: 0.0 secs
```

Tenemos bastantes menos reglas que antes.

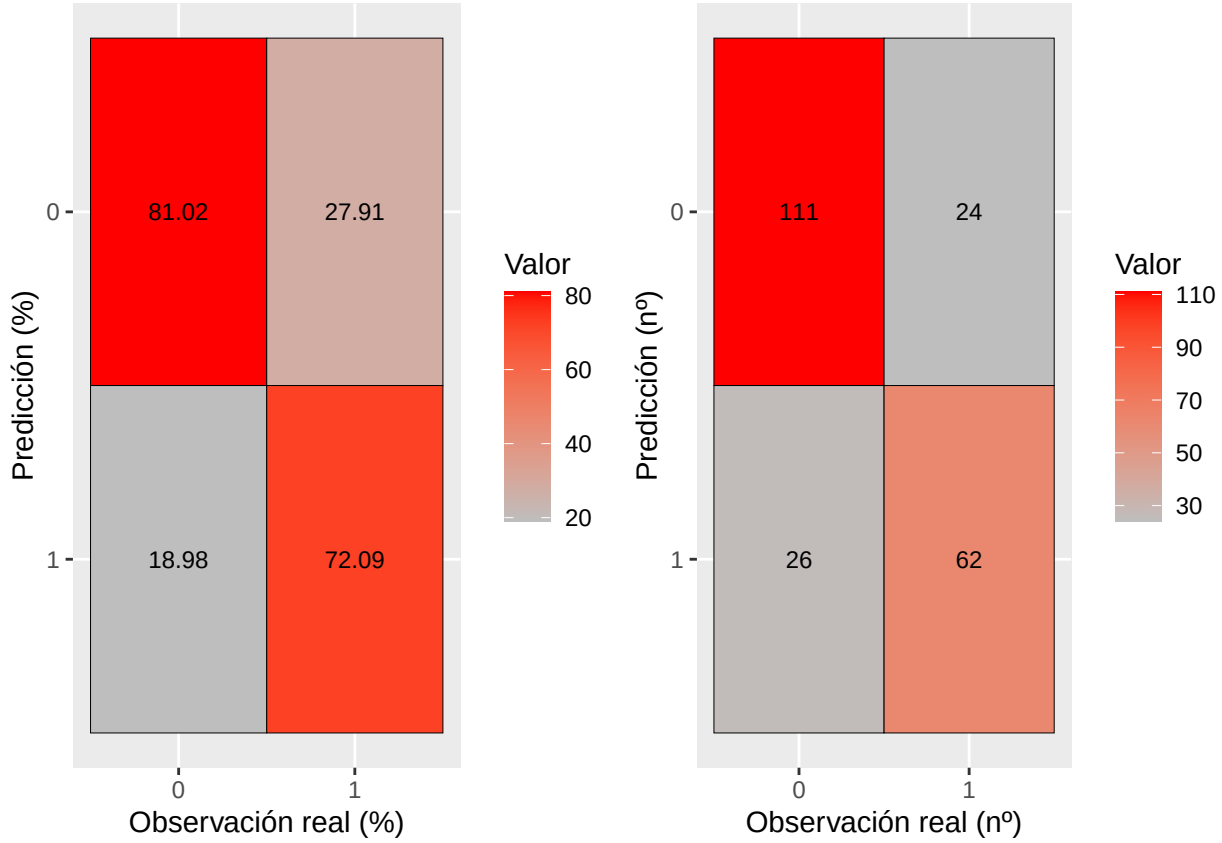
Vamos a ver cuantas variables ha utilizado ahora:






```
## [1] "La precisión del árbol C50 2 es: 77.5785 %"
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  223
##
##
##               | Prediction
##      Reality |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |       111 |        26 |       137 |
##              |     0.822 |     0.295 |           |
##              |     0.498 |     0.117 |           |
## -------------|-----------|-----------|-----------|
##            1 |        24 |        62 |        86 |
##              |     0.178 |     0.705 |           |
##              |     0.108 |     0.278 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       135 |        88 |       223 |
```

```
##              |     0.605 |     0.395 |           |
## -------------|-----------|-----------|-----------|
##
##
```



4.3.2.4.4 Tercer modelo C5.0 usando boosting La implementación del C5.0 dispone de la posibilidad de aplicar un *Boosting* similar a *AdaBoost*, que permite combinar varios árboles (técnica de *Ensemble*) para producir mejores modelos predictivos que si sólamente usasemos un árbol de decisión simple.

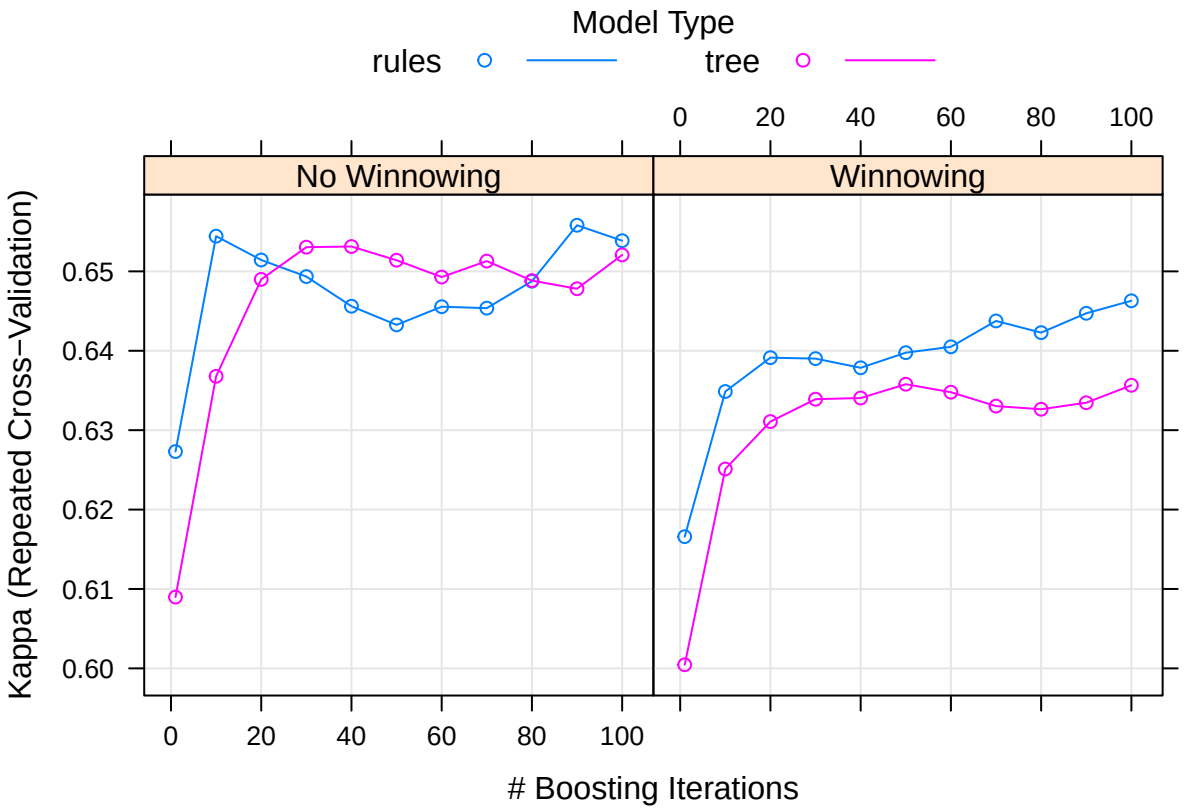
Para ello se utiliza el parámetro `trials`, en el que le indicamos cuantas iteraciones (que a la postre son árboles distintos) debe utilizar para generar el modelo *ensemble*.

Podríamos ir probando con distintos boosts variando el winnowing y los parámetros hasta encontrar el que mejor modelo nos generase. Es una labor tediosa que podemos simplificar usando la librería `caret`, que proporciona una forma de entrenar modelos modificando sus parámetros, vamos a usarlo para el C5.0 mostrando la comparativa posteriormente.

En este caso vamos a usar Cohen-Kappa como medición de calidad del modelo, y compararemos C5.0 generados con distintas iteraciones (boosting) combinándolo con winnowing y con rules/tree.

```
tuned <- train(trainX, trainy, method = "C5.0", tuneLength = 11,
 trControl = trainControl(method = "repeatedcv",
 repeats = 5),
 metric = "Kappa")
```

```
##  Accuracy     Kappa
## 0.8026906 0.5799658
```

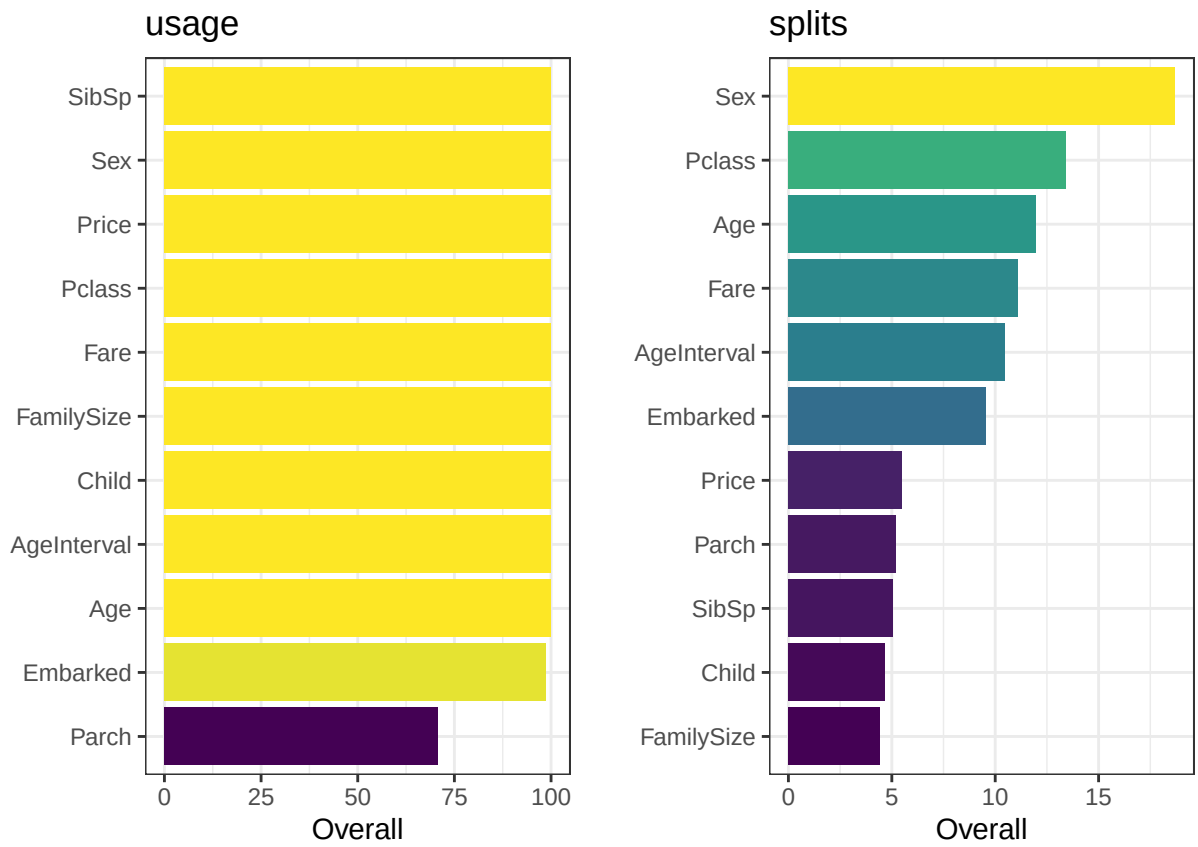


Los valores anteriores son orientativos y nos sirven para acotar más fácilmente los parámetros en los que buscar.

Vamos a crear el mejor árbol con winnowing y boost, usando 50 trials y sin reglas:

```
bc_boost_tree <- C5.0(
                      formula = Survived ~ .,
                      data    = data_train,
                      rules   = TRUE,
                      control = C5.0Control(seed = 123, winnow = FALSE),
                      trials = 30
                     )
```

Como antes, vemos la importancia de los atributos:

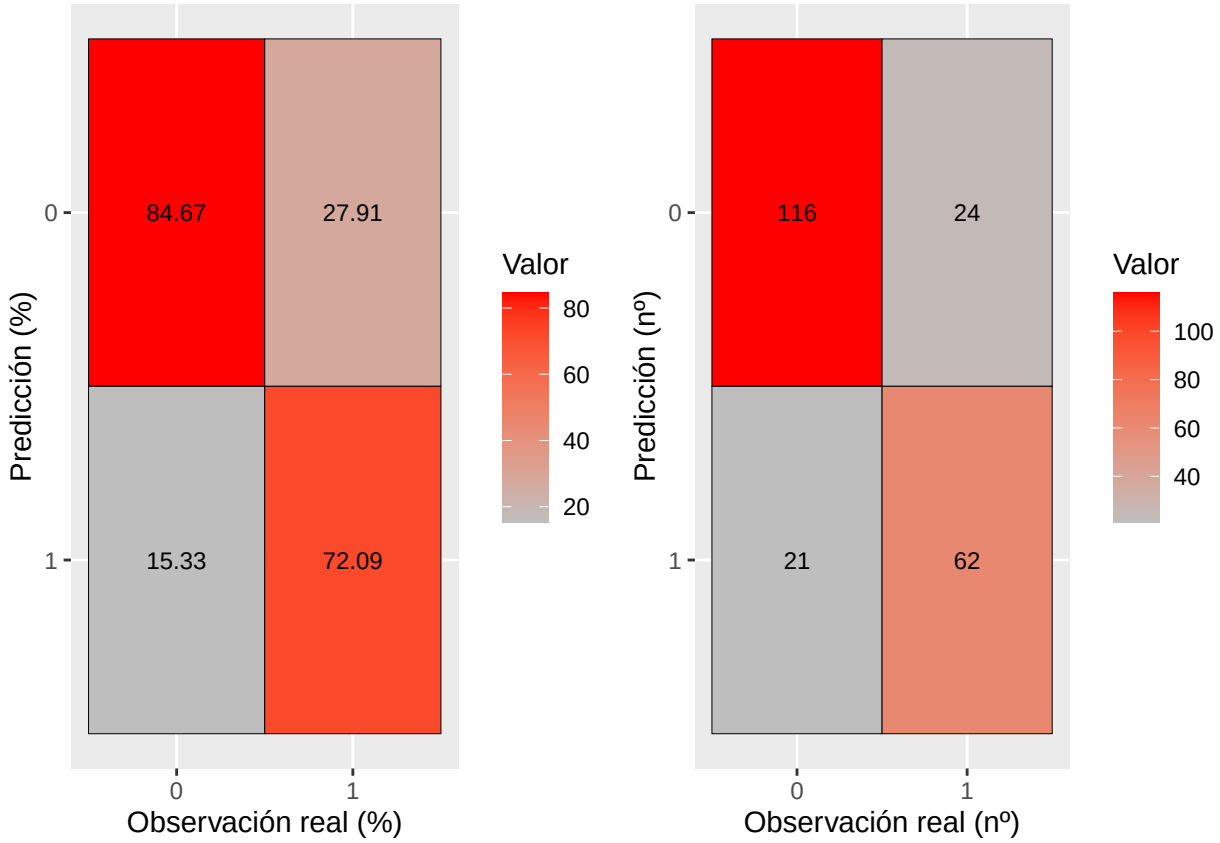


En este caso todos los atributos tienen un uso muy alto.

De nuevo calculamos matriz de confusión

```
## [1] "La precisión del árbol C50 3 es: 79.8206 %"
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  223
##
##
##              | Prediction
##      Reality |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |       116 |        21 |       137 |
##              |     0.829 |     0.253 |           |
##              |     0.520 |     0.094 |           |
## -------------|-----------|-----------|-----------|
##            1 |        24 |        62 |        86 |
##              |     0.171 |     0.747 |           |
##              |     0.108 |     0.278 |           |
```

```
## -------------|-----------|-----------|-----------|
## Column Total |       140 |        83 |       223 |
##              |     0.628 |     0.372 |           |
## -------------|-----------|-----------|-----------|
##
##
```



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

TODO: AMPLIAR ALGO

Durante la resolución de la práctica hemos aprovechado las capacidades didácticas del formato rmd para ir generando las gráficas y tablas en cada una de las secciones y explicando sus interpretaciones, esto es especialmente relevante en el caso del análisis exploratorio pues a la hora de tomar decisiones de limpieza y de uso de variables, disponer de las gráficas en el momento de la decisión es muy útil.

Con respecta a los modelos predictivos, se han ido también creando gráficas representando las tablas de confusión y extrayendo la precisión de los modelos. En los árboles se han mostrado, cuando eran suficientemente pequeños, las representaciones gráficas de los mismos, y cuando no se han mostrado datos y tablas numéricas explicando brevemente su utilidad.

Si que podemos exponer un resumen del comportamiento de los diferentes modelos predictivos creados a lo largo del ejercicio

```
knitr::kable(tabla.modelos)
```

Modelo	Precisión	Errores
glm1	79.372197309417	46
glm2	78.4753363228699	48
glm3	75.7847533632287	54
glm4	78.4753363228699	48
Decision Tree 1	77.5784753363229	50
Decision Tree 2	80.7174887892377	43
Random Forest	81.6143497757848	41
C50 1	78.9237668161435	47
C50 2	77.5784753363229	50
C50 3	79.8206278026906	45

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

TODO: los valores de precision cambian constantemente!!

Hemos procedido a realizar, sobre el conjunto de datos Titanic, las fases de limpieza, depuración y análisis de datos (eliminando variables innecesarias, imputando datos faltantes, comprobando supuestos estadísticos…), hasta finalmente implementar un conjunto de modelos de predicción basados en variables dadas y otras creadas nuevas.

El problema que nos planteábamos era: ¿Dado el conjunto de datos que disponemos, podemos predecir la supervivencia o no de un pasajero?.

Podemos diferenciar entre las conclusiones del análisis exploratorio y la capacidad de predecir de los modelos creados. Del análisis exploratorio podríamos responder a la pregunta: ¿Las mujeres y los niños primero? Y la respuesta es que parece que si, que tuvieron mayor proporción de supervivencia, y si ademas eras de clase rica (o viajabas en una clase mejor) tus probabilidades también aumentaban. Pero no podemos afirmar que fuese en todos los casos, como constata la dificultad para encontrar un modelo predictivo con una buena precisión.

Los diferentes modelos creados intentan resolverlo con resultados de precisión más bien discretos:

- Los modelos que mejor han funcionado han sido los árboles de decisión C5.0, y sobre todo el tercer árbol con boosting de 50 y winnowing con casi un 88% de precisión. No obstante utiliza todos los atributos y no es fácil de visualizar.

- Del resto de modelos planteado destacaríamos tres de ellos con precisión de 83,78%: Dos regresiones logísticas usando datos enriquecidos con variables sintéticas, y un árbol de decisión con profundidad 5 para el que las variables más significativas resultaron ser las siguientes: **Sexmale**, **Pclass3** y **FamilySize** (aunque también hemos visto que hay otras variables influyentes en los modelos, y que eran de esperar como **Age** y sus variables derivadas, como **Child** y **AgeInterval** y que suponen también una lógica).

También hemos comprobado que la regresión logística con interacción de variables (modelo glm4) mejora la predicción de supervivientes (a costa de fallar algo más en los no supervivientes). Aunque la precisión total de ambos modelos fue equivalente, en el modelo de regresión logística ha fallado en que fallecen el 11,68% y el modelo predice que sobreviven (y sobreviven el 23.53% y el modelo predice lo contrario), mientras que en el árbol (tree2) se ha fallado en lo siguiente, fallecen realmente un 7,30% y el modelo predice que sobreviven (y sobreviven el 30,59% y el modelo predice que mueren).

Creemos que es preferible que se equivoque más en la predicción de que no van a sobrevivir y al final si lo hacen que en lo contrario (que se prediga que sobrevive y luego no lo haga) Esto parece tener sentido, ya que si observamos las 2 variables más importantes del árbol, **Sexmale** (hombres) y **Pclass3** (3a clase), precisamente son las variables que suponen el mayor índice de mortalidad.

6.1 Exportación de los datos

Vamos a exportar dos ficheros que estarán disponibles en el repositorio github: - Un fichero "titanic_final.csv" que contiene las 12 variables con las que se ha trabajado, con la imputación de los datos realizada, con las variables no utilizadas del dataset origen eliminadas, y con las transformaciones que hemos considerado necesarias.

- Un fichero "titanic_predict.csv", en el que además de las columnas que tiene el fichero anterior, tiene una columna nueva llamada "Survived_Predicted", que es la columna con la predicción de la variable "Survived" realizada con el modelo escogido.

TODO##### Exportación de los datos

7 Tabla de contribuciones

Contribuciones	Firma
Investigación Previa	Alexis Germán Arroyo Peña, Gabriel Pulido de Torres
Redacción de las respuestas	Alexis Germán Arroyo Peña, Gabriel Pulido de Torres
Desarrollo código	Alexis Germán Arroyo Peña, Gabriel Pulido de Torres

8 Bibliografía y referencias

[1] Arboles de decision, Random Forest, Gradient Boosting y C5.0. [en línea], [sin fecha]. [Consulta: 2 enero 2021]. Disponible en: https://www.cienciadedatos.net/documentos/33_arboles_decision_random_forest_gradient_boosting_c50#C50.

[2] JARMAN, K.H., 2013. The Art of Data Analysis: How to Answer Almost Any Question Using Basic Statistics. S.l.: s.n. ISBN 9781118413357.

[3] MAT, L.S., OSWALDO, D., MIREIA, T. y GONZ, C., [sin fecha]. Introducción a la limpieza y análisis de los datos. ,

[4] MOLINA, L.C. y SANGÜESA I SOLÉ, R., [sin fecha]. Módulo 8: Evaluación de modelos. ,