Tipología y ciclo de vida de los datos

Práctica 2

Fernando Rodríguez López

13/5/2019

Descripción del dataset

El dataset seleccionada es el del hundimiento del titanic de Kaggle [https://www.kaggle.com/c/titanic/data]

El hundimiento del **RMS Titanic** es una de los hundimientos de barcos más famosos de la historia. El incidente ocurrió entre el día 14 y 15 de abril de 1912. Durante su viaje inaugural entre Southampton y Nueva York, el transatlántico británico cochó contra un iceberg en el oceano Atlántico frente a las costas de Terranova. Tras el choque el translatlántico se hundío y murieron 1502 personas de 2224 pasajeros y tribulates.

Esta tragedia ha sido una de las mayores tragedias naúticas en tipo de paz. Las causas del número de fallecidos fueron consecuencia de la falta de botes salvavidas. Pero además, en diferentes estudios se ha visto que la suerte de los supervivientes estaban realionadas con distintas características de los viajes y tripulantes.

En el siguiente estudio se pretende ver que tipo de personas tuvieron la suerte de sobrevivir. Teniendo en cuenta su género, clase social y edad.

Los datos nos los dan divididos en dos grupos:

- **El conjunto de entrenamiento** usado para crear el modelo de entrenamiento para un modelo. Para este grupo se le aporta la clase de salida (también conocidad como *ground truth*)
- **El conjunto de test** Normalmente usado para comprobar lo bien que predice el modelo, pero como no se aporta la clase de salida.No podemos utilizarlo para la comprobación del modelo, sino que se utiliza como respuesta para la competición de kaggle.

Conjunto de entrenamiento

El conjunto de entrenamiento es un fichero csv en código ASCII que consta de los siguiente atributos. Este fichero incluye las cabeceras dentro del fichero y los campos están separados por ",".

Variable	Descripción	Valores
PassengerId	Identificador de pasajero	
Survived	Sobrevivió	0 = No, 1 = Sí
pclass	Tipo del billete	1 = Primera clase, 2 = Segunda Clase, 3 = Tercera Clase
Name	Nombre	
Sex	Género	male = Hombre, female= Mujer
Age	Edad en Años	
Sibsp	Número de familiares a bordo (hermanos, pareja)	
Parch	Número de famliares a bordo (padres e hijos)	
Ticket	Número del billete	
Fare	Precio del billete	
Cabin	Número de cabina	

Variable	Descripción	Valores
Embarked	Puerto de embarque	C = Cherbourg, Q = Queenstown, S = Southampton

Conjuto de test

El conjuto de test también es un fichero csv en código ASCII que consta de los siguientes atributos Este fichero incluye las cabeceras dentro del fichero y los campos están separados por ",".

Variable	Descripción	Valores
PassengerId	Identificador del pasajero	
Pclass	Tipo del billete	1 = Primera clase, 2 = Segunda Clase, 3 = Tercera Clase
Name	Nombre	
Sex	Género	male = Hombre, female= Mujer
Age	Edad en Años	
SibSp	Número de familiares a bordo (hermanos, pareja)	
Parch	Número de famliares a bordo (padres e hijos)	
Ticket	Número del billete	
Fare	Precio del billete	
Cabin	Número de cabina	
Embarked	Puerto de embarque	C = Cherbourg, Q = Queenstown, S = Southampton

Para una mejor compresión del dataset tenemos que tener encuenta las siguientes consideraciones

Age: la edad en caso de viajeros que no superen más de un año es fraccional.

SibsP: Determina el número de familiares del tipo hermanos y pareja - Hermanos: incluye hermanos, hermanas, hermanástros y hermanástras - Pareja: esposos y esposas. Los novios y amantes fueron descartados **Parch**: - Padre: madre y padre - Hijo: hijos, hijas, hijastros e hijastras.

Integración y selección de los datos de interés a analizar

El primer paso que vamos a realizar es la carga de ambos ficheros en un mismo dataframe. Como podemos comprobar los dos ficheros, tienen los mismos campos exceptuando la clase de salida, que en el caso de conjunto test no existe. Ya que es el objeto de la competición de Kaggle. Pero uniendo los dos ficheros en un dataframe único, podemos realizar un análisis y limpieza única con toda la población, observando datos perdidos, valores extremos y otros posibles errores. Una vez realizado el trabajo de limpieza, podeos volver a separar para aplicar los modelos.

Hay que tener en cuenta que el archivo csv debe estar en el directorio "kaggle" dentro de nuestro directorio de trabajo. En caso contrario hay que especificar la ruta absoluta al archivo.

```
# Leemos los datos de entrenamiento
train <- read.csv("./kaggle/train.csv")
# Leemos los datos de test
test <- read.csv("./kaggle/test.csv")

# Variable con las propiedades no incluyendo la clase salida
properties = colnames(test)
```

```r
# Variable con la clase salida
class = c("Survived")
# Creamos un dataframe unico con todos los datos
titanic_raw <- bind_rows(train, test)

# Creamos un dataframe donde realizamos las operaciones
titanic <- titanic_raw
```

Realizamos una comprobación visual, para ver si se han cargado los datos con las propiedades que hemos determinado en el apartado anterior.

```r
# Echamos un vistazo a los datos
str(titanic)
```

```
## 'data.frame':    1309 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        : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
##  $ Age        : num  22 38 26 35 35 NA 54 2 27 14 ...
##  $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket     : chr  "A/5 21171" "PC 17599" "STON/O2. 3101282" "113803" ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : chr  "" "C85" "" "C123" ...
##  $ Embarked   : chr  "S" "C" "S" "S" ...
```

Observamos que hay 1309 que son la suma de los 418 elementos de test más los 891 elementos de entrenamiento que corresponde con la información que nos aporta kaggle.

Clase de salida Survived

Todas estas observaciones tiene 12 propiedades, que corresponde a 11 atributos más la clase de salidad *Survived* donde los datos de test tendrían que tener el valor de NA.

Pero pasamos a comprobarlo.

```r
# Número de instancias con el valor Survived Nulo
str(titanic %>% filter(is.na(Survived)))
```

```
## 'data.frame':    418 obs. of  12 variables:
##  $ PassengerId: int  892 893 894 895 896 897 898 899 900 901 ...
##  $ Survived   : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ Pclass     : int  3 3 2 3 3 3 3 2 3 3 ...
##  $ Name       : chr  "Kelly, Mr. James" "Wilkes, Mrs. James (Ellen Needs)" "Myles, Mr. Thomas Fran
##  $ Sex        : Factor w/ 2 levels "female","male": 2 1 2 2 1 2 1 2 1 2 ...
##  $ Age        : num  34.5 47 62 27 22 14 30 26 18 21 ...
##  $ SibSp      : int  0 1 0 0 1 0 0 1 0 2 ...
##  $ Parch      : int  0 0 0 0 1 0 0 1 0 0 ...
##  $ Ticket     : chr  "330911" "363272" "240276" "315154" ...
##  $ Fare       : num  7.83 7 9.69 8.66 12.29 ...
##  $ Cabin      : chr  "" "" "" "" ...
##  $ Embarked   : chr  "Q" "S" "Q" "S" ...
```

```r
#Comprobamos  que los PassengerID son los mismos en el dataframe titanic con Survived a NA y los de t
str(setdiff(test %>% select("PassengerId"), titanic %>% filter(is.na(Survived)) %>% select("Passenger
```

```
## 'data.frame':    0 obs. of  1 variable:
```

```
##  $ PassengerId: int
```

Como vemos el número de observaciones con Survived igual a NA corresponde al número de test y además no hay diferencias de los códigos de los pasajeros (PassengerId). Por lo que los NA corresponde a los datos del conjunto de test.

Así que hemos realizado correctamente la integración de los dos ficheros csv.

```
titanic$Survived <- as.factor(titanic$Survived)
levels(titanic$Survived)
```

```
## [1] "0" "1"
```

PassengerId

Ahora procedemos ha imprimir un resumen del dataframe para estudiar nuestra propiedades

```
# Resumen de las propiedades sin contar la clase de salida
summary(titanic[properties])
```

```
##   PassengerId       Pclass          Name              Sex
##  Min.   :   1   Min.   :1.000   Length:1309        female:466
##  1st Qu.: 328   1st Qu.:2.000   Class :character   male  :843
##  Median : 655   Median :3.000   Mode  :character
##  Mean   : 655   Mean   :2.295
##  3rd Qu.: 982   3rd Qu.:3.000
##  Max.   :1309   Max.   :3.000
##
##       Age            SibSp            Parch          Ticket
##  Min.   : 0.17   Min.   :0.0000   Min.   :0.000   Length:1309
##  1st Qu.:21.00   1st Qu.:0.0000   1st Qu.:0.000   Class :character
##  Median :28.00   Median :0.0000   Median :0.000   Mode  :character
##  Mean   :29.88   Mean   :0.4989   Mean   :0.385
##  3rd Qu.:39.00   3rd Qu.:1.0000   3rd Qu.:0.000
##  Max.   :80.00   Max.   :8.0000   Max.   :9.000
##  NA's   :263
##       Fare            Cabin             Embarked
##  Min.   :  0.000   Length:1309        Length:1309
##  1st Qu.:  7.896   Class :character   Class :character
##  Median : 14.454   Mode  :character   Mode  :character
##  Mean   : 33.295
##  3rd Qu.: 31.275
##  Max.   :512.329
##  NA's   :1
```

El campo **PassengerId** es únicamente para identificar a cada uno de los pasajeros. Por lo que no formará parte de ninguno de nuestro estudios. Pero lo asignamos como el valor de **id** de nuestro Dataframe.

```
# Asignamos el identificador de dataframe con los valores de PassengerId
rownames(titanic) <- titanic$PassengerId
# Eliminamos de la variable properties la variable
#titanic$PassengerId <- NULL
properties <- properties[!properties %in% "PassengerId"]
```

Pclass

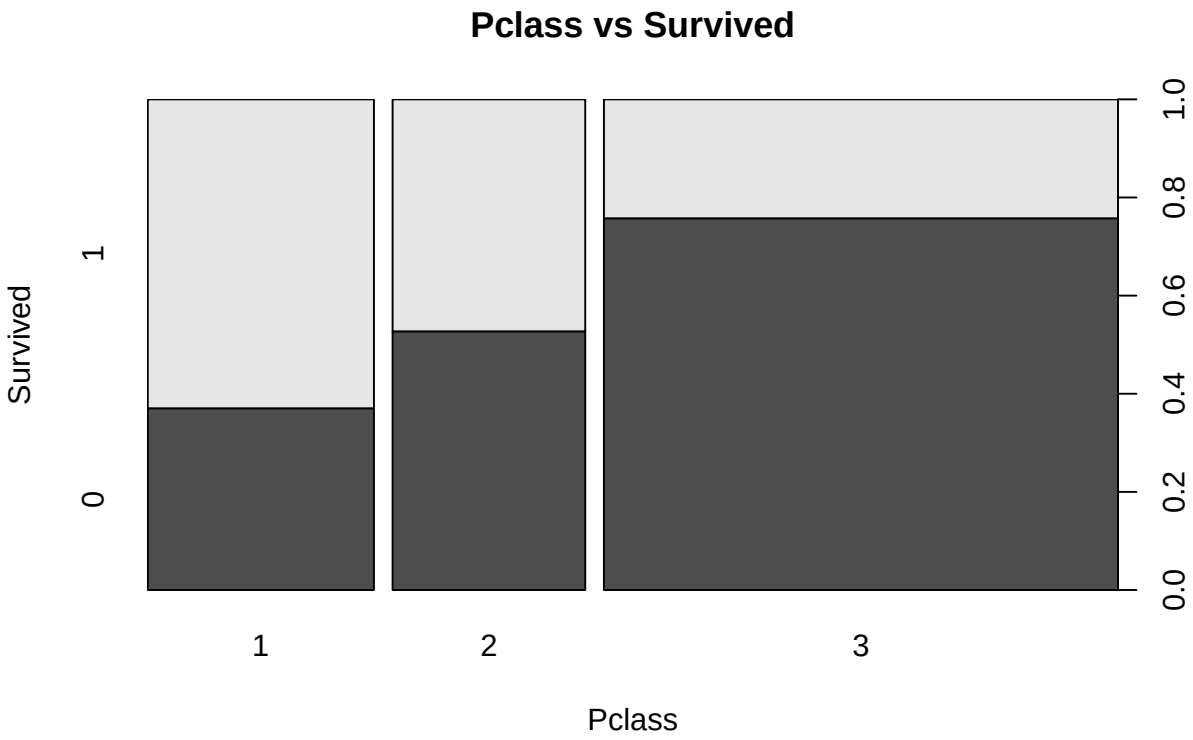
Vemos que la propiedad Pclass es numérica y debería de ser factor ya que no representa una categoricación numérica, además no tiene ningún valor perdido.

```
titanic$Pclass <- factor(titanic$Pclass)
# Viajeros según la clase
  local({
   .Table <- with(titanic, table(Pclass))
   cat("\ncounts:\n")
   print(.Table)
   cat("\npercentages:\n")
   print(round(100*.Table/sum(.Table), 2))
   })
```

```
##
## counts:
## Pclass
##   1   2   3
## 323 277 709
##
## percentages:
## Pclass
##     1     2     3
## 24.68 21.16 54.16
```

Si representamos esta categoría de clase frente a la clase de salida, podemos observar datos interesantes.

```
with(titanic, plot(Pclass,Survived, xlab="Pclass", ylab="Survived" ,main ="Pclass vs Survived"))
```



Los viajeros de la clase 1 tienen mucha más probabilidad de sobrevivir que el resto de clases. Los viajeros de la clase 2 tienen un 50 % de sobrevivir y los viajeros de las clase 3 tiene mucha mayor probabilidad de no sobrevivir.

Por lo tanto, parece que la variable clase puede ser determinante para predecir si una persona sobrevive o no.

Name -> Título

Revisando visualmente el campo **Name**(nombre) observamos que están los títulos de cada uno de los viajeros. Es decir si son señores, señoras, señoritas. Lo cual podría ser variable diferenciadora para determinar si se puede salvar o no.

Para ellos sacaremos el Título según los nombres

```
# Cogemos los títulos según los nombres
titanic$Title <- gsub('(.*, )|(\\..*)', '', titanic$Name)
# Presentamos los anteriores títulos enfrentados al género
table(titanic$Sex, titanic$Title)
```

```
##
##          Capt Col Don Dona  Dr Jonkheer Lady Major Master Miss Mlle Mme
##   female    0   0   0    1   1        0    1     0      0  260    2   1
##   male      1   4   1    0   7        1    0     2     61    0    0   0
##
##           Mr Mrs  Ms Rev Sir the Countess
##   female   0 197   2   0   0            1
##   male   757   0   0   8   1            0
```

Procedemos a convertir los títulos obtenidos en un grupo más reducido

```
# Titulos que vamos a convertir a Mr
toMr_title  <- c ('Don', 'Major', 'Capt', 'Jonkheer', 'Rev', 'Col', 'Sir')
# Convertirmos dichos títulos a Mr
titanic$Title[titanic$Title %in% toMr_title]  <- 'Mr'
# Titulos que vamos a convertir a Mrs
toMrs_title  <- c('the Countess', 'Mme', 'Dona', 'Lady')
# Convertirmos dichos títulos a Mr
titanic$Title[titanic$Title %in% toMrs_title]  <- 'Mrs'
# Titulos que vamos a convertir a Miss
toMiss_title  <- c('Mlle', 'Ms')
# Convertirmos dichos títulos a Miss
titanic$Title[titanic$Title %in% toMiss_title]  <- 'Miss'

# Convertimos los Dr - female en Mrs
titanic$Title[(titanic$Title %in% "Dr") & titanic$Sex == "female"] <- "Mrs"
# Convertimos los Dr - male en Mr
titanic$Title[(titanic$Title %in% "Dr") & titanic$Sex == "male"] <- "Mr"

# Añadimos  el atributo Title
properties <- append(properties, "Title")
# Show title counts by sex again
table(titanic$Sex, titanic$Title)
```

```
##
##          Master Miss  Mr Mrs
##   female      0  264   0 202
##   male       61    0 782   0
```

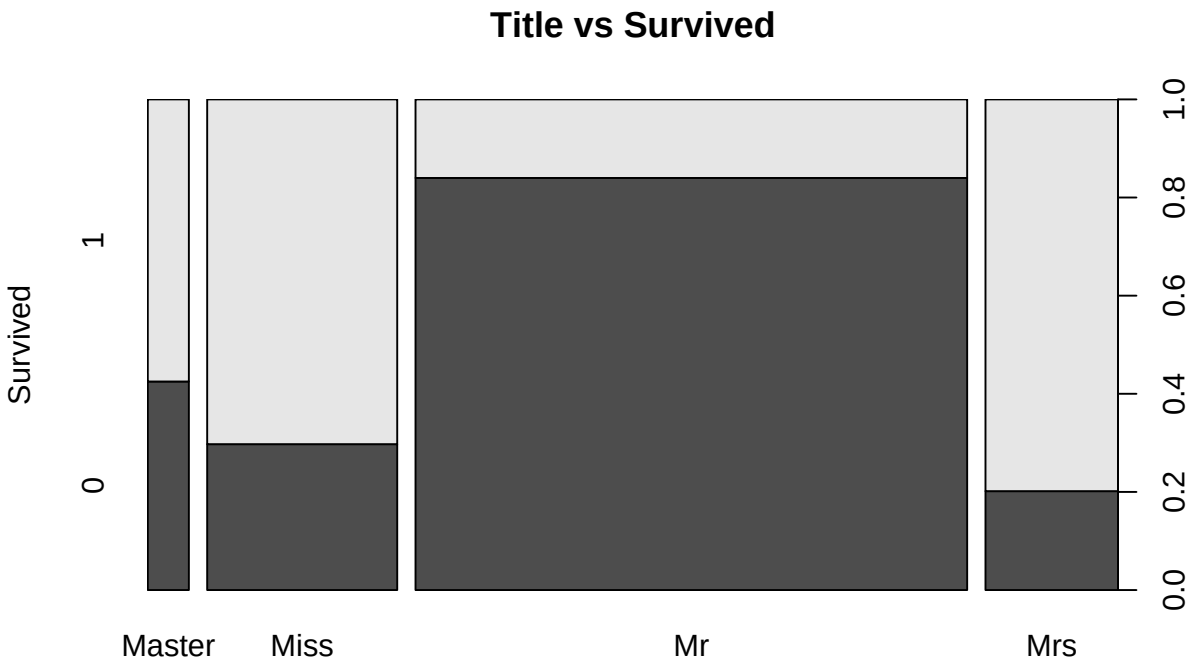
```
# Convertimos el campo en factor

titanic$Title <- as.factor(titanic$Title)
```

```
# Viajeros según la Titulo
  local({
   .Table <- with(titanic, table(Title))
   cat("\ncounts:\n")
```

```
   print(.Table)
   cat("\npercentages:\n")
   print(round(100*.Table/sum(.Table), 2))
   })
```

```
##
## counts:
## Title
## Master   Miss     Mr    Mrs
##     61    264    782    202
##
## percentages:
## Title
## Master   Miss     Mr    Mrs
##   4.66  20.17  59.74  15.43
```

```
with(titanic, plot(Title,Survived, xlab="Title", ylab="Survived", ,main ="Title vs Survived"))
```



En la gráfica anterior, donde enfrentamos el título con la salida, observamos que tanto la Mrs, como Miss tienen una álta probabilidad de sobrevivir. Como se ve esto nos puede hacer pensar que las mujeres (females) van a tener mucha más probabilidad que los hombres de sobrevivir. Lo cuál se verán en el apartado siguiente. Y entre los hombres, observamos que los Mr. tienen mayor probabilidad de no sobrevivir, mientras que los que tiene el título de Master se aproximan a los procentajes de probabilidad de supervivencia de las mujeres.

En espera de los resultados, que veamos al analizar la varible Sex, a priori este campo puede resultar interesante para nuestra predicción, incluyo que la combinación con Sex, puede determinar bastante la supervivencia o no de un pasajero.

Pero eliminamos el campos **Name** que no parece útil para ninguno de los posibles modelos.

```
# Eliminamos de la variable properties la variable
#titanic$Name <- NULL
properties <- properties[!properties %in% "Name"]
```

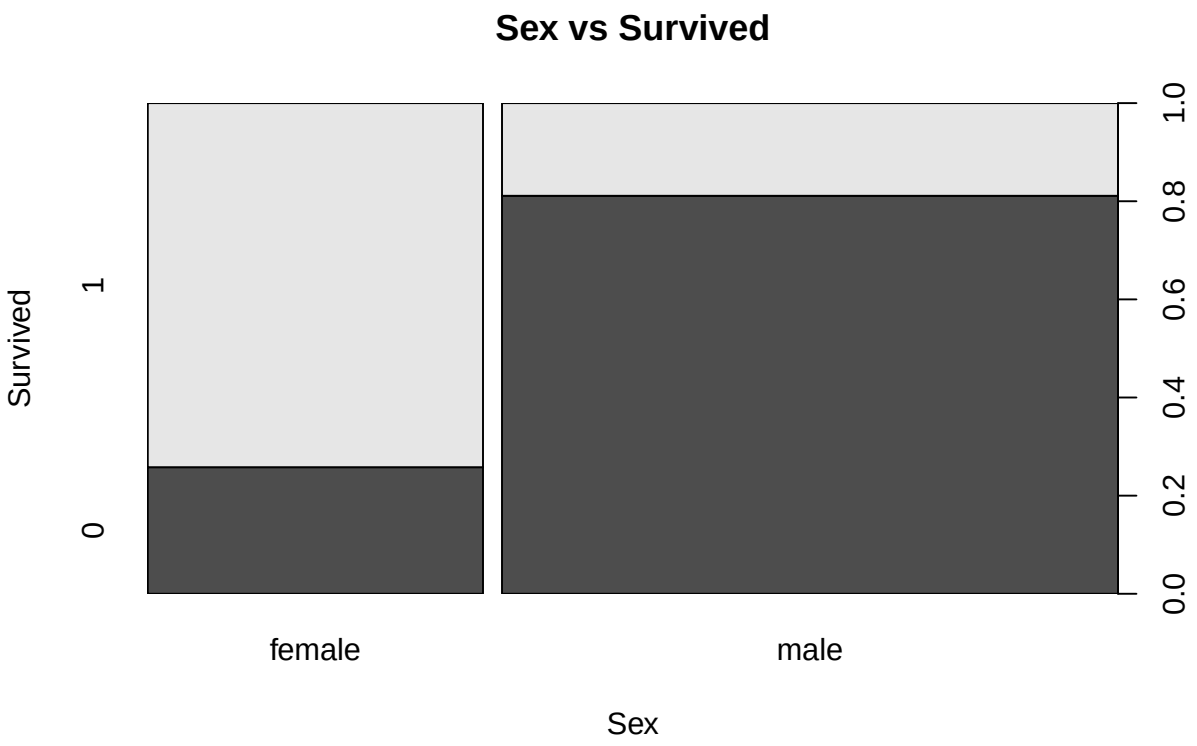
Sex

El campo **Sex**(género) podría ser útil para nuestros modelos como hemos visto anteriormente con los datos del título, que indirectamente establece el género del viajero.

```
# Viajeros según el género
  local({
   .Table <- with(titanic, table(Sex))
   cat("\ncounts:\n")
   print(.Table)
   cat("\npercentages:\n")
   print(round(100*.Table/sum(.Table), 2))
   })
```

```
##
## counts:
## Sex
## female   male
##    466    843
##
## percentages:
## Sex
## female   male
##   35.6   64.4
```

```
with(titanic, plot(Sex,Survived, xlab="Sex", ylab="Survived", main ="Sex vs Survived"))
```



Al enfrentrar el género con la supervivencia, observamos que las mujeres al igual que ocurria en el caso de los títulos de las mujeres tienen un alto procentaje de supervivencia y por el contrario los hombre un alto porcentaje de no sobrevivivir.

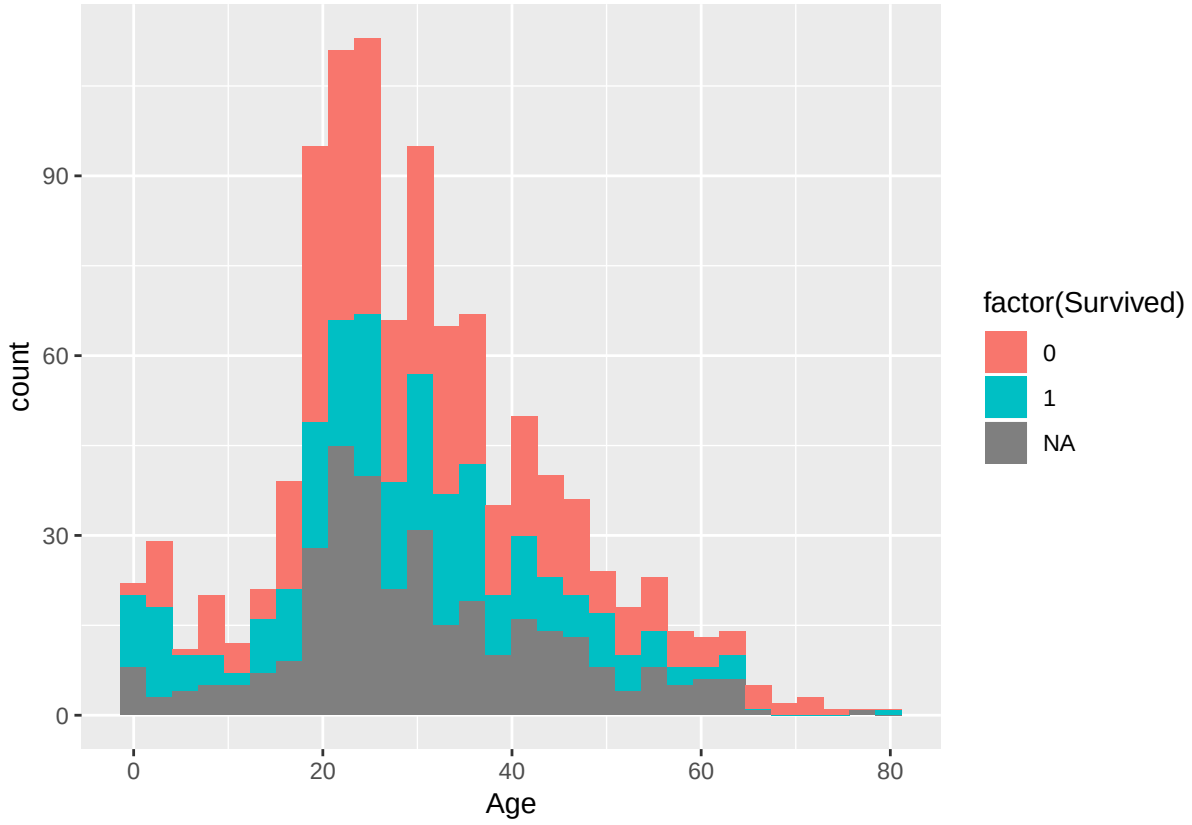
Parece razonable, que la combinación de *Sex* y *Title* nos podría ayudar a determinar con una porcentaje bastate exacto la clase final en un arbol de decisión.

Age

El campo **Age**(edad) podría ser útil para nuestros modelos por lo que lo mantenemos, pero vemos que tiene valores perdídos que estudiaremos en el siguiente apartado.

```
ggplot(titanic, aes(Age, fill = factor(Survived))) +
  geom_histogram(bins=30)
```

```
## Warning: Removed 263 rows containing non-finite values (stat_bin).
```



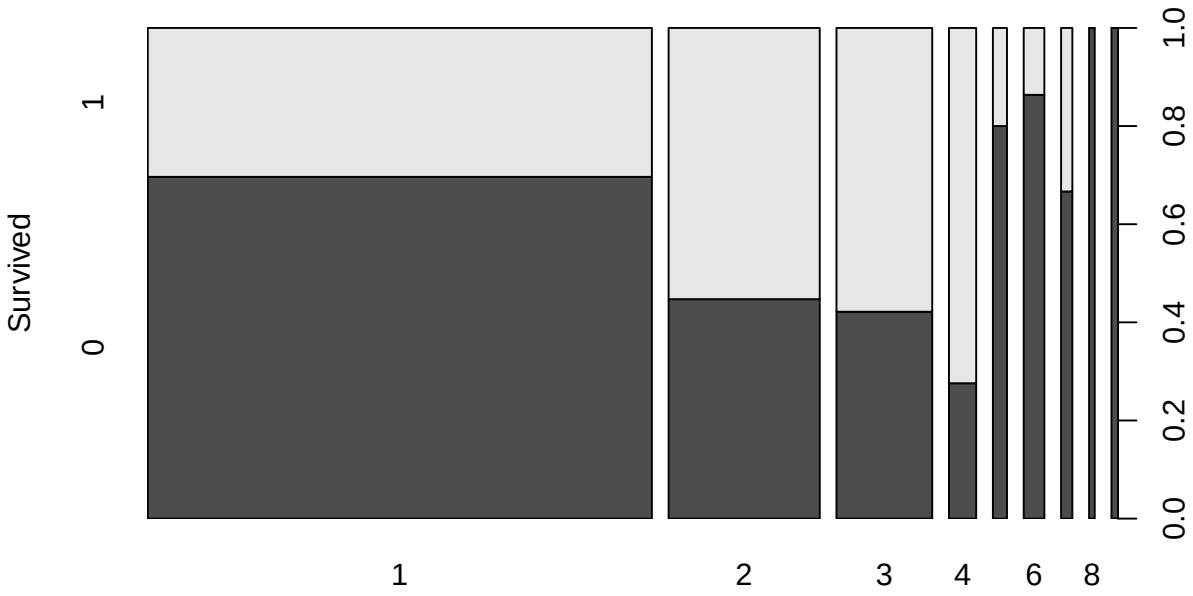
Sibsp, Parch -> Family

Los dos siguientes atributos **Sibsp**(hermanos, pareja) y **Parch** (padres e hijos) pueden ser interesantes para nuestros modelos, pero creemos que podría ser válido para nuestros modelos la unión de los dos en un nuevo campo que sea **Familiy**.

```
titanic$Family <- titanic$SibSp + titanic$Parch + 1
properties <- append(properties, "Family")
```

Es un atributo numérico, pero se puede considerar también cualitativo.

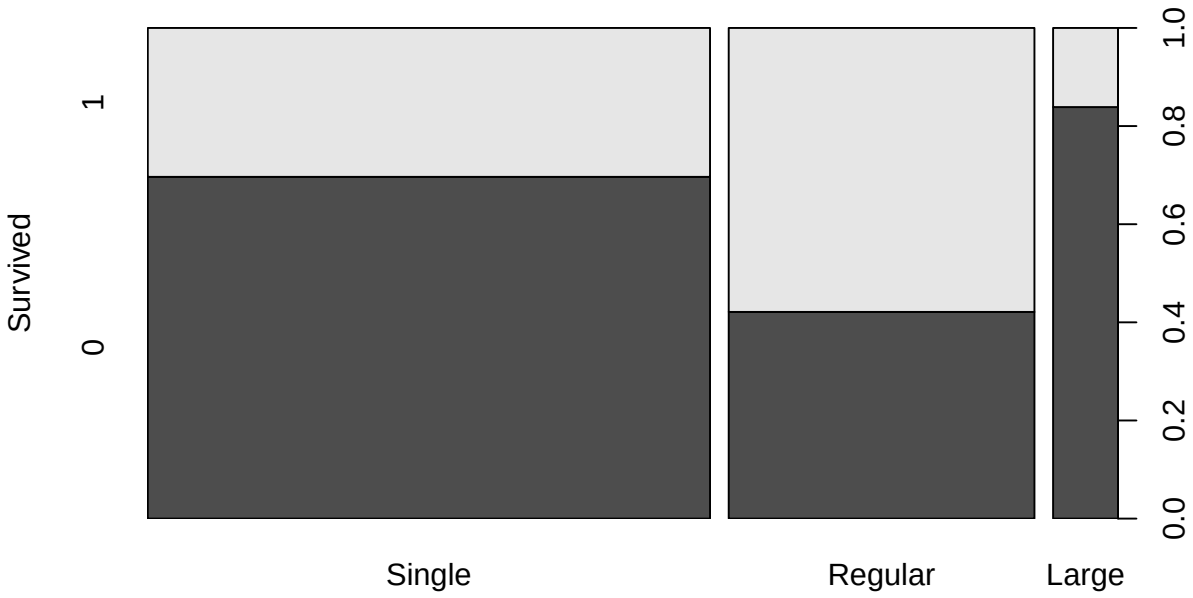
```
titanic$Family <- as.factor(titanic$Family)
with(titanic, plot(Family,Survived, xlab="Family", ylab="Survived",  main ="Family vs Survived"))
```



Como vemos en la gráfica, podemos observar que las familias con más 5 o más miembros tienen mucha probabilidad de no sobrevivir. Por otra parte los solteros también tiene alta probabilidad de no sobrevivir y sin embargo las familias con ménos de 5 miembros tienen alta probabilidad de sobrevivir.

Por ellos vamos a realizar una agrupación al campo **FamilyType** según estas agrupaciones.

```
titanic$Family <- as.numeric(titanic$Family)
titanic$FamilyType <- 'Regular'
titanic$FamilyType[titanic$Family ==1]  <- 'Single'
titanic$FamilyType[titanic$Family >=5]  <- 'Large'

titanic$FamilyType <- ordered(titanic$FamilyType, c("Single", "Regular", "Large"))

titanic$FamilyType <- as.factor(titanic$FamilyType)
with(titanic, plot(FamilyType,Survived, xlab="Family Type", ylab="Survived", main ="Family Type vs Su
```



Este campo nuevo parece bastante interesante para poder discernir si un viajero tiene posibilidad de sobrevivir o no.

De esta última gráfica, se puede observar que las famílias entre 2 y 4 miembros tiene mayor probabilidad de sobrevivir. Miembras que los solteros y las famílias numerosas de más de 5 miembros su probabiliadd de sobrevivir es mucho más baja. Por lo que parece que el tipo de familia puede ser interesante para discernir la clase de salida.

```
# Añadimos este campo a la properties
properties = c(properties, "FamilyType")
# Eliminar Family, SibsSPy Parch
properties <- properties[!properties %in% c("Family", "SibSp", "Parch")]
```

Ticket

El campo *Ticket* está como tipo characters, aunque no parece un campo útil, para nuestro modelo, pero vamos a convertirlo en factor, para ver si puede ser útil.

```
titanic$Ticket <- as.factor(titanic$Ticket)
# Hacemos un sumary
summary(titanic)
```

```
##   PassengerId   Survived   Pclass      Name              Sex
##  Min.   :   1   0   :549   1:323   Length:1309        female:466
##  1st Qu.: 328   1   :342   2:277   Class :character   male  :843
##  Median : 655   NA's:418   3:709   Mode  :character
##  Mean   : 655
##  3rd Qu.: 982
##  Max.   :1309
##
##       Age            SibSp           Parch            Ticket
##  Min.   : 0.17   Min.   :0.0000   Min.   :0.000   CA. 2343:  11
##  1st Qu.:21.00   1st Qu.:0.0000   1st Qu.:0.000   1601    :   8
##  Median :28.00   Median :0.0000   Median :0.000   CA 2144 :   8
```

```
##  Mean   :29.88   Mean   :0.4989   Mean   :0.385   3101295 :   7
##  3rd Qu.:39.00   3rd Qu.:1.0000   3rd Qu.:0.000   347077  :   7
##  Max.   :80.00   Max.   :8.0000   Max.   :9.000   347082  :   7
##  NA's   :263                                      (Other) :1261
##       Fare            Cabin             Embarked            Title
##  Min.   :  0.000   Length:1309        Length:1309        Master: 61
##  1st Qu.:  7.896   Class :character   Class :character   Miss  :264
##  Median : 14.454   Mode  :character   Mode  :character   Mr    :782
##  Mean   : 33.295                                         Mrs   :202
##  3rd Qu.: 31.275
##  Max.   :512.329
##  NA's   :1
##      Family        FamilyType
##  Min.   :1.000   Single :790
##  1st Qu.:1.000   Regular:437
##  Median :1.000   Large  : 82
##  Mean   :1.867
##  3rd Qu.:2.000
##  Max.   :9.000
##
```

```
titanic %>%
    group_by(Ticket) %>%
    count()
```

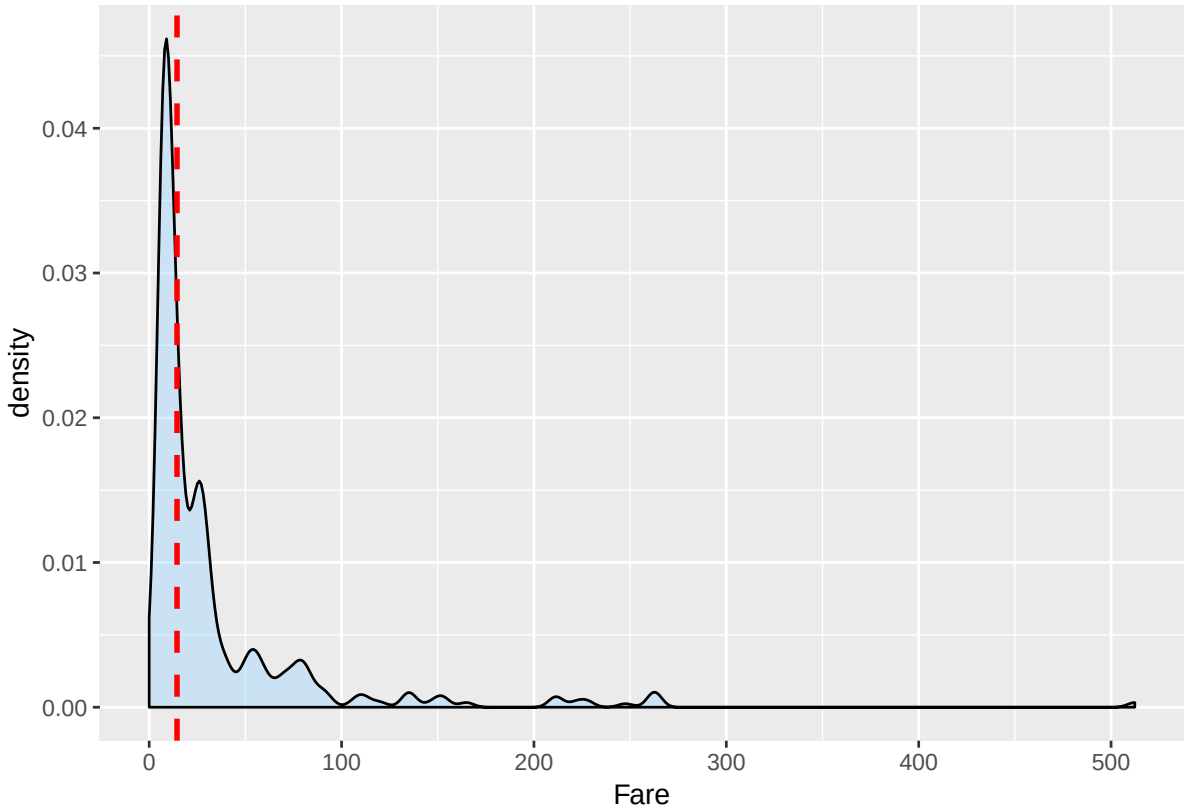
```
## # A tibble: 929 x 2
## # Groups:   Ticket [929]
##    Ticket     n
##    <fct>  <int>
##  1 110152     3
##  2 110413     3
##  3 110465     2
##  4 110469     1
##  5 110489     1
##  6 110564     1
##  7 110813     2
##  8 111163     1
##  9 111240     1
## 10 111320     1
## # ... with 919 more rows
```

Como podemos observar de los 1309 hay 1261 tipos distintos de Tickets, por lo tanto no parece un campo muy relevante y lo eliminamos de nuestro dataframe.

```
# Eliminamos de la variable properties la variable
#titanic$Ticket <- NULL
properties <- properties[!properties %in% "Ticket"]
```

Fare

EL campo **Fare**(precio del billete) a priori parece interesante para un modelo de predicción de si el pasajero sobrevive o no. Vemos que tiene un valor perdido que también veremos en el próximo apartado.



Al observar la función de densidad observamos que hay muchos datos extremos, y nos hace plantearnos que el campo Fare es el precio del billete completo y puede estar definido por el número de personas de dicho ticket. Lo analizaremos cuando veamos los valores extremos.

Cabin -> Deck

EL campo **Cabin** (nombre del camarote) al igual que pasaba con Ticket no parece muy interesante para los modelos, pero vamos a factorizar.

```
titanic$Cabin <- as.factor(titanic$Cabin)
titanic %>%
    group_by(Cabin) %>%
    count()
```

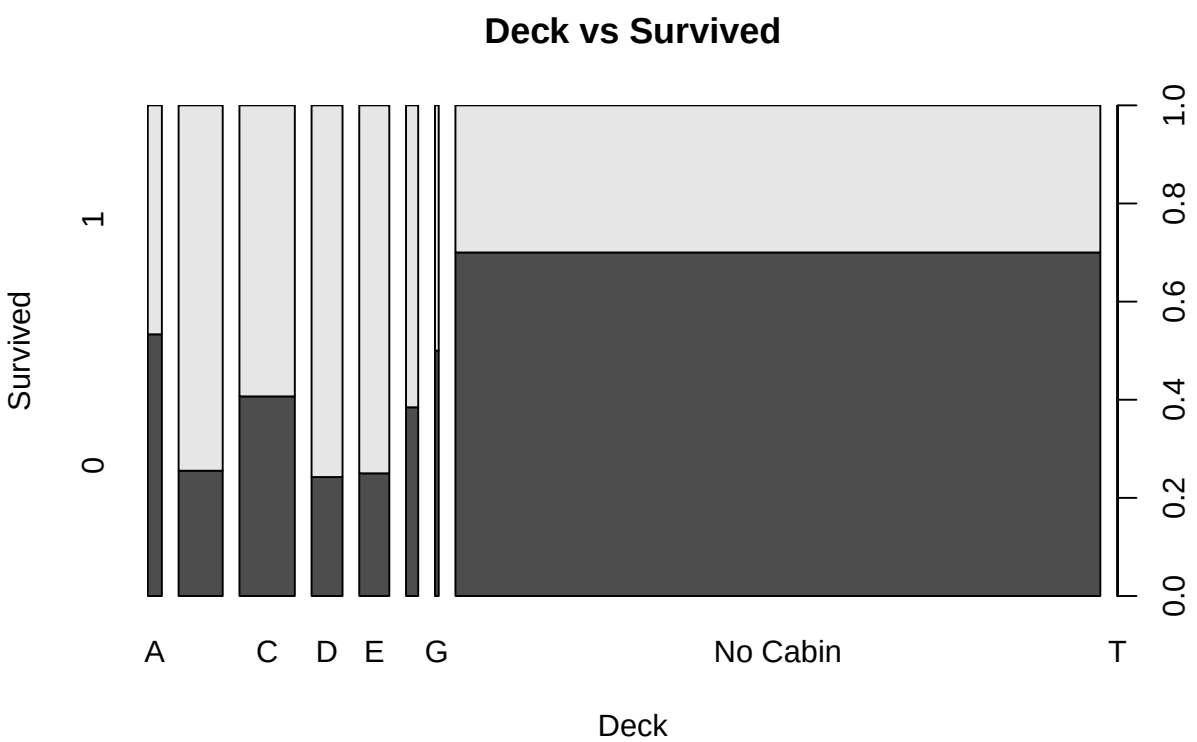
```
## # A tibble: 187 x 2
## # Groups:   Cabin [187]
##    Cabin     n
##    <fct> <int>
##  1 ""     1014
##  2 A10       1
##  3 A11       1
##  4 A14       1
##  5 A16       1
##  6 A18       1
##  7 A19       1
##  8 A20       1
##  9 A21       1
## 10 A23       1
## # ... with 177 more rows
```

```
# Hacemos un sumary
summary(titanic[properties])
```

```
##  Pclass      Sex           Age              Fare
##  1:323   female:466   Min.   : 0.17   Min.   :  0.000
##  2:277   male  :843   1st Qu.:21.00   1st Qu.:  7.896
##  3:709                Median :28.00   Median : 14.454
##                       Mean   :29.88   Mean   : 33.295
##                       3rd Qu.:39.00   3rd Qu.: 31.275
##                       Max.   :80.00   Max.   :512.329
##                       NA's   :263     NA's   :1
##              Cabin        Embarked           Title       FamilyType
##                 :1014   Length:1309        Master: 61   Single :790
##  C23 C25 C27    :   6   Class :character   Miss  :264   Regular:437
##  B57 B59 B63 B66:   5   Mode  :character   Mr    :782   Large  : 82
##  G6             :   5                      Mrs   :202
##  B96 B98        :   4
##  C22 C26        :   4
##  (Other)        : 271
```

En el resumen vemos que hay más de 271 tipos de cabinas, por lo que parecería interesante ya que se agruparían muchos pasajeros, pero uno de los grupos contiene 1014 pasajeros. Por esto parece que no es muy interesante. Pero podemos agruparlos por las cubiertas de la cabina, para ver si es interesante esta nueva propiedad.

```
titanic$Cabin <- as.character(titanic$Cabin)
titanic$Deck<-sapply(titanic$Cabin, function(x) strsplit(x, NULL)[[1]][1])
titanic$Deck[is.na(titanic$Deck)] <- "No Cabin"
titanic$Deck <- as.factor(titanic$Deck)
with(titanic, plot(Deck,Survived, xlab="Deck", ylab="Survived", main ="Deck vs Survived"))
```



Este propiedad parece más interesante, porque hay una probabilidad de 70% que si un pasajero no tuviera cabina, no sobreviviese. Mientas que si tiene cabina la probabiliad baja dependiendo de la cubierta.

```
# Eliminamos de la variable properties la variable
#titanic$Cabin <- NULL
```

```
properties <- properties[!properties %in% "Cabin"]
# Añadimos la variable Deck
properties <- c(properties, "Deck")
```

Embarked

El último campo **Embarked**(puerto de embarque) es de tipo texto y lo pasamos a factor para ver si puede resultar interesante.

```
titanic$Embarked <- as.factor(titanic$Embarked)
titanic %>%
    group_by(Embarked) %>%
    count()
```

```
## # A tibble: 4 x 2
## # Groups:   Embarked [4]
##   Embarked     n
##   <fct>    <int>
## 1 ""           2
## 2 C          270
## 3 Q          123
## 4 S          914
```

De la agrupación vermos que tenemos 4 niveles y uno de ello es valor perdido, que estudiaremos en el próximo apartado.

```
# Viajeros según el embarque
  local({
   .Table <- with(titanic, table(Embarked))
   cat("\ncounts:\n")
   print(.Table)
   cat("\npercentages:\n")
   print(round(100*.Table/sum(.Table), 2))
   })
```

```
##
## counts:
## Embarked
##       C   Q   S
##   2 270 123 914
##
## percentages:
## Embarked
##           C     Q     S
##  0.15 20.63  9.40 69.82
```

```
with(titanic, plot(Embarked,Survived, xlab="Embarked", ylab="Survived", main ="Embarked  vs Survived"
```

De la gráfica podemos observar que parece que dependiendo de donde se realizase el embarque, hay variación de la probabilidad de sobrevivir por lo que parece un campo interesante de estudio.

Aunque la diferencia de probabilidad, no parece muy determinante ya que están muy próximas las probabilidades.

Limpieza de los datos.

Valores vacios o que continen 0

Como hemos visto en el apartado anterior de nuestras propiedades numéricas tenemos valores nulos en **Age** y **Fare** y de tipo factor en *Embarked*.

Valor *Fare* con valor NA

Buscamos el único valor que contiene NA en su propiedad *Fare*

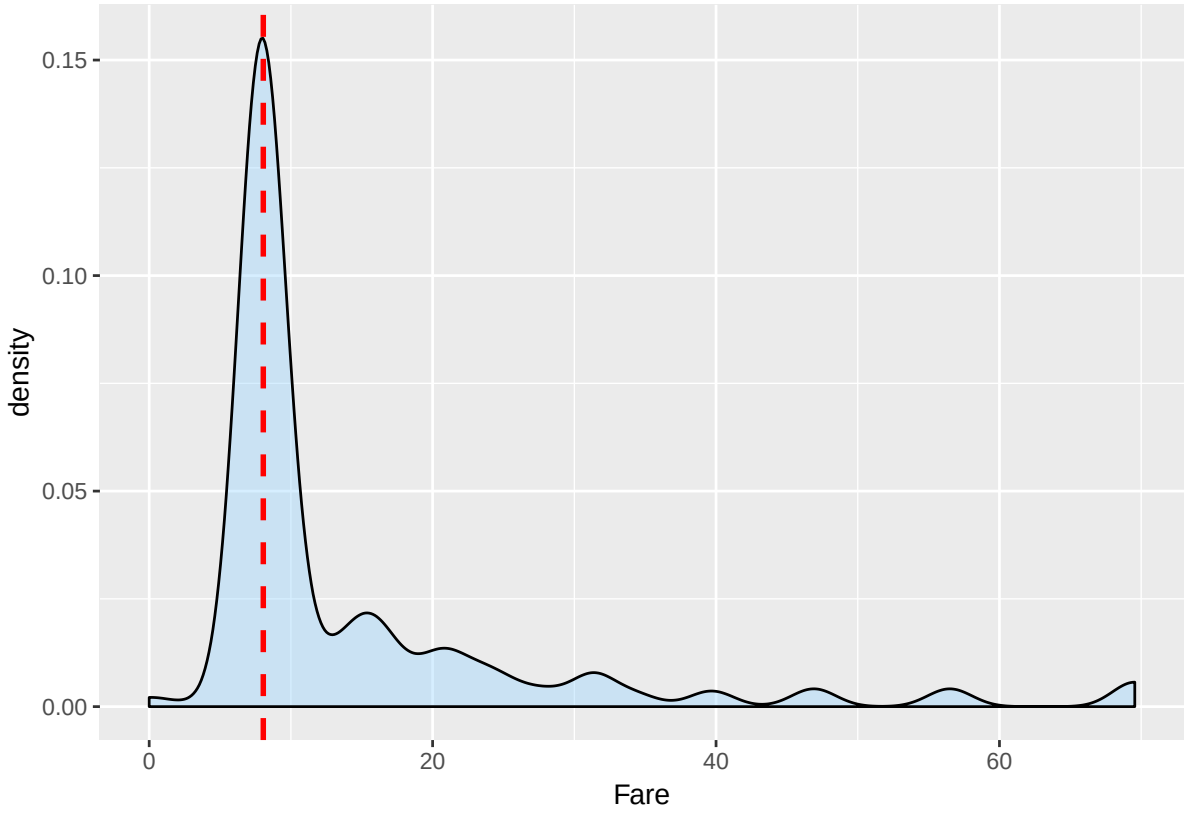
```
titanic %>% filter(is.na(titanic$Fare))
```

```
##   PassengerId Survived Pclass               Name  Sex  Age SibSp Parch
## 1        1044     <NA>      3 Storey, Mr. Thomas male 60.5     0     0
##   Ticket Fare Cabin Embarked Title Family FamilyType     Deck
## 1   3701   NA              S    Mr      1     Single No Cabin
```

De este pasajero observamos que su embarque fué en *Southampton* ('S') y es de tercera clase, que parece propiedades que determinarían el precio del embarque.

```
ggplot(titanic[titanic$Pclass == '3' & titanic$Embarked == 'S', ],
  aes(x = Fare)) +
  # Función de densidad de los valores de Fare filtrados
  geom_density(fill = '#99d6ff', alpha=0.4, na.rm=T) +
  # Dibujamos la recta de la mediana
```

```
  geom_vline(aes(xintercept=median(Fare, na.rm=T)),
    colour='red', linetype='dashed', lwd=1)
```



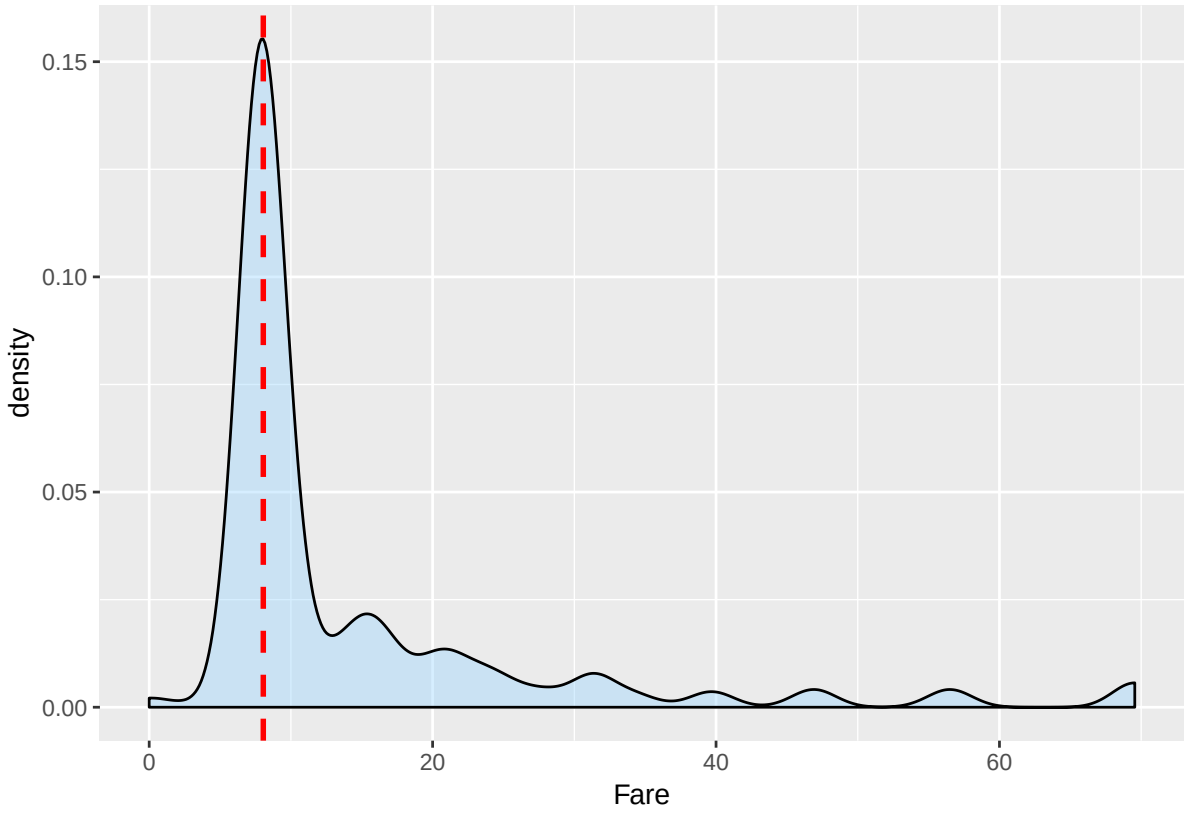
De esta visualización vemos que la mayoría de los valores se concentran cerca de la mediana, por lo que parece razonable sustituir el valor perdido con el valor de la mediana del grupo que corresponde con la misma clase y el mismo embarque.

```
# Reemplazamos el valor perdido con el valor de la mediana
titanic$Fare[1044] <- median(titanic[titanic$Pclass == '3' & titanic$Embarked == 'S', ]$Fare, na.rm =
sprintf ("Valor Fare reemplazado: %s", titanic$Fare[1044])
```

```
## [1] "Valor Fare reemplazado: 8.05"
```

Si representamos de nuevo nuestras función densidad, vemos que es bastante similar.

```
ggplot(titanic[titanic$Pclass == '3' & titanic$Embarked == 'S', ],
  aes(x = Fare)) +
  # Función de densidad de los valores de Fare filtrados
  geom_density(fill = '#99d6ff', alpha=0.4) +
  # Dibujamos la recta de la mediana
  geom_vline(aes(xintercept=median(Fare)),
    colour='red', linetype='dashed', lwd=1)
```



Valor *Age* con valor NA

Como hemos visto los valores perdidos del atributo *Age* es de 263 que frente al total suponen un 20% que es una gran cantidad de valores perdidos.

```
summary(titanic %>% select(properties) %>% filter(is.na(Age)))
```

```
##  Pclass      Sex           Age           Fare        Embarked    Title
##  1: 39   female: 78   Min.   : NA   Min.   :  0.00    :  0    Master:  8
##  2: 16   male  :185   1st Qu.: NA   1st Qu.:  7.75   C: 58    Miss  : 51
##  3:208                Median : NA   Median :  8.05   Q: 73    Mr    :177
##                       Mean   :NaN   Mean   : 19.82   S:132    Mrs   : 27
##                       3rd Qu.: NA   3rd Qu.: 22.80
##                       Max.   : NA   Max.   :227.53
##                       NA's   :263
##    FamilyType        Deck
##  Single :200   No Cabin:240
##  Regular: 48   C       :  8
##  Large  : 15   D       :  4
##                A       :  3
##                E       :  3
##                F       :  3
##                (Other) :  2
```

Al ser un gran úmnero de valores, no podemos permitirnos elmininar dichos datos.

Para ello tenemos que imputar los posibles valores. Para ellos utilizaremos dos modelos uno el K vecinos y otro con un Random-forest según la biblioteca mice orientada para obtener rellenear valores vacios.

Primero con el KNN de la libería VIM.

```
# La función kNN genera una nueva columna lógica que
# indica si se han imputado valores o no
mod_knn <- kNN(titanic, variable = ("Age"))
```

Con un Random Forest con la librería mice.

```
seed = 129
set.seed(seed)
mice_mod <- mice(titanic[, !names(titanic) %in% c('PassengerId','Name','Ticket','Cabin','Survived')],
```

```
##
##  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_output <- complete(mice_mod)
```

Después de obtener los valores, con los dos métodos, representamos la función densidad, y la comparamos con los datos originales. Para valorar, como varía la función densidad de los datos con las imputaciones realizadas.

```
# Función densidad de la Edad con los datos original
Age_original <- ggplot(titanic,
  aes(x = Age)) +
  # Función de densidad de los valores de Age filtrados
  geom_density(fill = '#99d6ff', alpha=0.4, na.rm=T) +
  # Dibujamos la recta de la mediana
  geom_vline(aes(xintercept=median(Age, na.rm=T)),
    colour='red', linetype='dashed', lwd=1)
# Función densidad de la Edad con los datos completados con Knn
Age_knn <- ggplot(mod_knn,
  aes(x = Age)) +
  # Función de densidad de los valores de Age filtrados
  geom_density(fill = '#99d600', alpha=0.4, na.rm=T) +
  # Dibujamos la recta de la mediana
```

```
  geom_vline(aes(xintercept=median(Age, na.rm=T)),
    colour='red', linetype='dashed', lwd=1)

# Función densidad de la Edad con los datos completados con Random-Forest según la libería mice

Age_rf <- ggplot(mice_output,
  aes(x = Age)) +
  # Función de densidad de los valores de Age filtrados
  geom_density(fill = '#ff0f55', alpha=0.4, na.rm=T) +
  # Dibujamos la recta de la mediana
  geom_vline(aes(xintercept=median(Age, na.rm=T)),
    colour='red', linetype='dashed', lwd=1)

figure <- ggarrange(Age_original, Age_knn, Age_rf,
                    labels = c("Original", "Knn", "Random-Forest"),
                    ncol = 1, nrow =3)
figure
```

De la gráficas, observamos como el método **Random-Forest** obtiene una gráfica de densidad de la Edad muy parecida a la muestra original sin tener en cuenta los valores perdidos y la mediana no varía Sin embargo, con el método **Knn** obtenemos una gráfica más distorsionada e incluso la mediana se desplaza un poco. Por lo que procedemos a remplazar en nuestro dataframe los datos obtenidos con el método **Random-Forest** en los valores perdidos

```
# Reemplazamos los datos de la edad en nuestro dataframe original
titanic["Age"] <- mice_output$Age
```

Valor *Embarked* con valor vacio

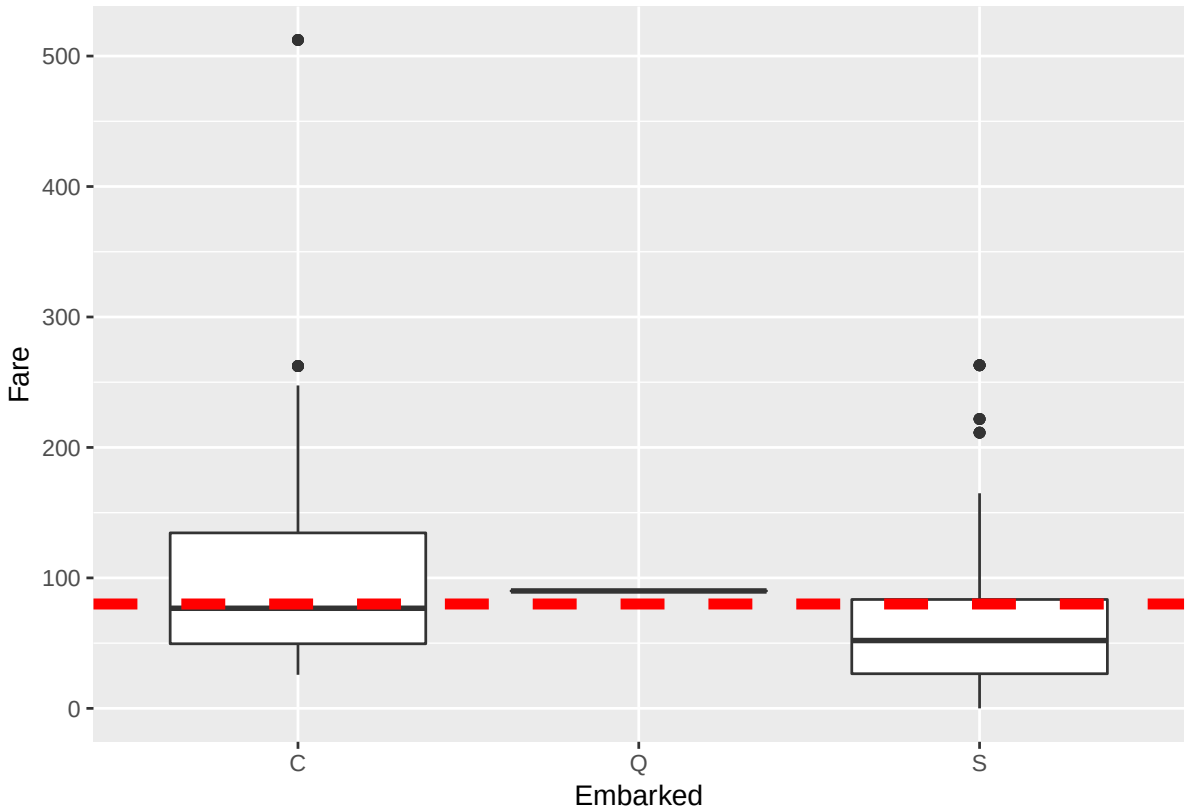
Presentamos los valores con embarque vacio

```
titanic %>% filter(Embarked == "")
```

```
##   PassengerId Survived Pclass                                     Name
## 1          62        1      1                      Icard, Miss. Amelie
## 2         830        1      1 Stone, Mrs. George Nelson (Martha Evelyn)
##      Sex Age SibSp Parch Ticket Fare Cabin Embarked Title Family
## 1 female  38     0     0 113572   80   B28           Miss      1
## 2 female  62     0     0 113572   80   B28            Mrs      1
##   FamilyType Deck
## 1     Single    B
## 2     Single    B
```

Observamos que las instancias que tienen el embarque vacio son de la Clase 1 y tienen un precio de embarque de 80. Para ver como se distribuyen los precios de los embarques representamos los *boxplot* de la población según los embarques, descartando los elementos que tienen embarque vacio

```
# Eliminamos de la población los que tiene embarque vacio
embark_fare <- titanic %>%
  filter(PassengerId != 62 & PassengerId != 830 & Pclass==1)
# Repesentamos los boxplot y una línea roja con el valor del precio del pasaje de los valores perdido
ggplot(embark_fare, aes(x = Embarked, y = Fare)) +
  geom_boxplot() +
  geom_hline(aes(yintercept=80),
    colour='red', linetype='dashed', lwd=2)
```



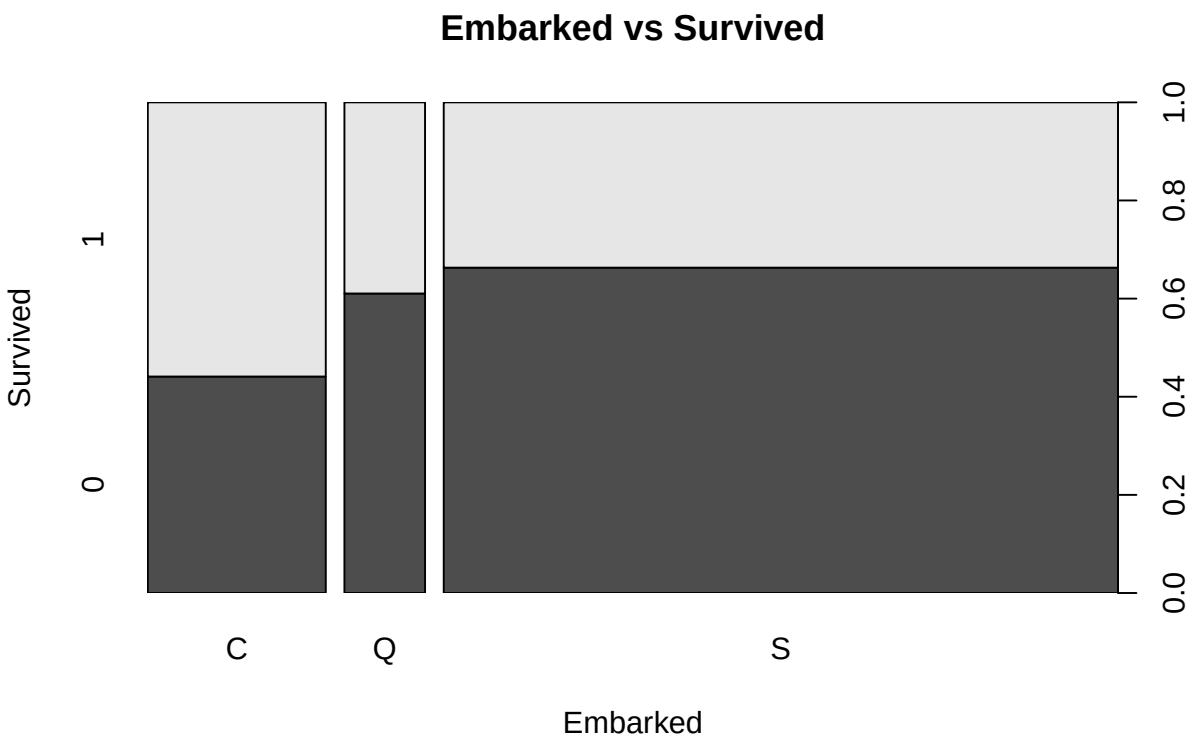
Como vemos la mediana de un embarque en Charbourg ('C') de primera clase coincide con el precio de 80 de la instancia que desconocemos el embarque por lo que parece razonable reemplazarlo por el valor de Charbourg

```
titanic$Embarked[titanic$Embarked==""] <- "C"
titanic$Embarked <- as.factor(as.character(titanic$Embarked))
```

Al volver a dibujar los tipos de embarques respecto a la probabilidad de Sobrevivir observamos que no

se varía la probabilidad de la población de sobrevivir al realizar la modificación.

```
with(titanic, plot(Embarked,Survived, xlab="Embarked", ylab="Survived", main ="Embarked vs Survived"))
```



Aunque como hemos visto la probabilida de salida no es muy determinante, pero si que puede ayudar en algunos casos donde haya dudas con la probabilidad de otras de las varibles.

Identificación y tratamiento de valores externos

Para detectar la presencia de valores atípicos examinaremos primero el resumen de los cinco números de Tukey, donde podremos observar un análisis descriptivo de los datos

Para obtener los datos sólo utilizaremos las variables numéricas **Age**, **Fare**, y la calculada **Family** a partir de **SibSp**, **Parch**.

```
numeric_properties <- c ("Age",  "Fare", "Family")
summary(titanic %>% select(numeric_properties))
```

```
##       Age             Fare             Family
##  Min.   : 0.17   Min.   :  0.000   Min.   :1.000
##  1st Qu.:21.00   1st Qu.:  7.896   1st Qu.:1.000
##  Median :28.00   Median : 14.454   Median :1.000
##  Mean   :29.62   Mean   : 33.276   Mean   :1.867
##  3rd Qu.:38.00   3rd Qu.: 31.275   3rd Qu.:2.000
##  Max.   :80.00   Max.   :512.329   Max.   :9.000
```

Los cinco números también se representan gráficamente con **boxplot**

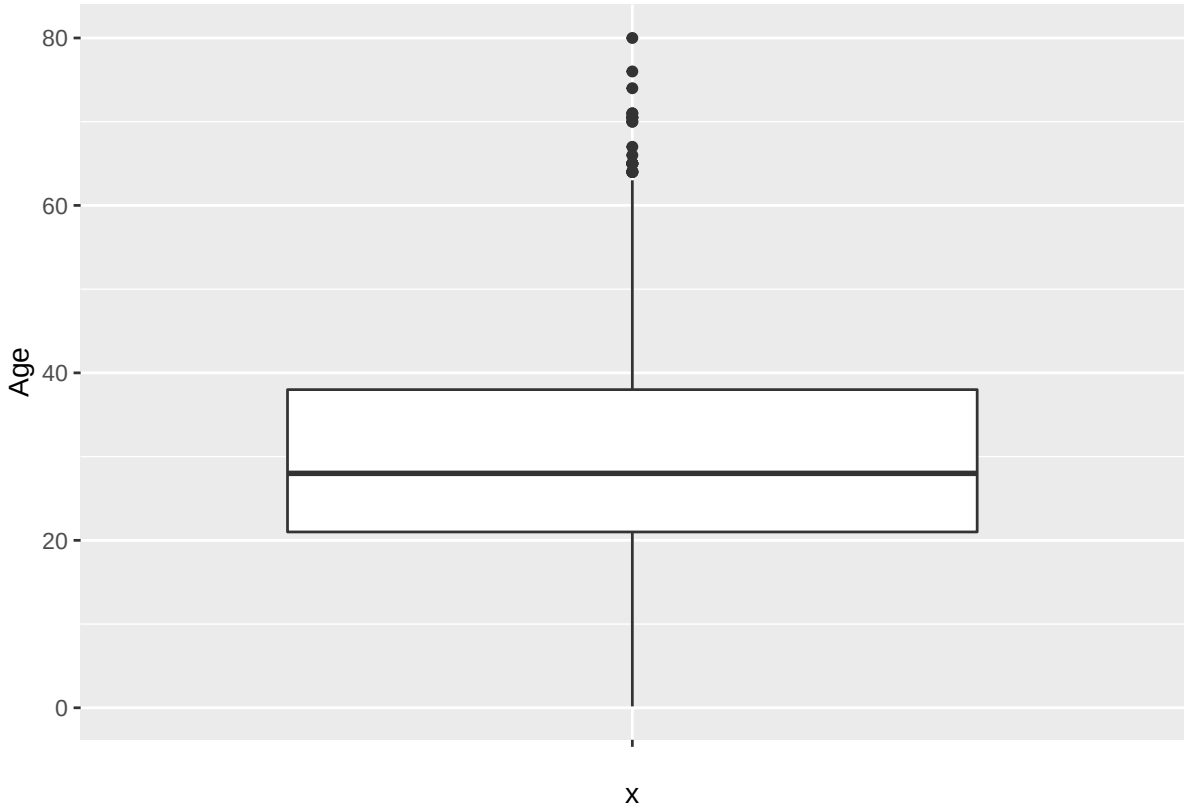
```
sapply(titanic[numeric_properties], boxplot.stats)
```

```
##       Age        Fare        Family
## stats Numeric,5  Numeric,5   Numeric,5
## n     1309       1309        1309
## conf  Numeric,2  Numeric,2   Numeric,2
## out   Numeric,26 Numeric,171 Numeric,125
```

Age

Para estudiar los valores extremos dibujamos el boxplot de la propiedad

```
Age_boxplot <- ggplot(titanic, aes(x="", y=Age) ) +
  geom_boxplot()
Age_boxplot
```



De la gráfica, observamos que la mayoría de la población se encuentra entre 0 y 60 años aproximadamente. Pero hay pasajeros que se encuentra entre los 60 y los 80 años. Por lo que no parece que hay errores tipográficos, y parecen valores razonables ya que no hay ninguna edad que pueda ser considerada erronea.

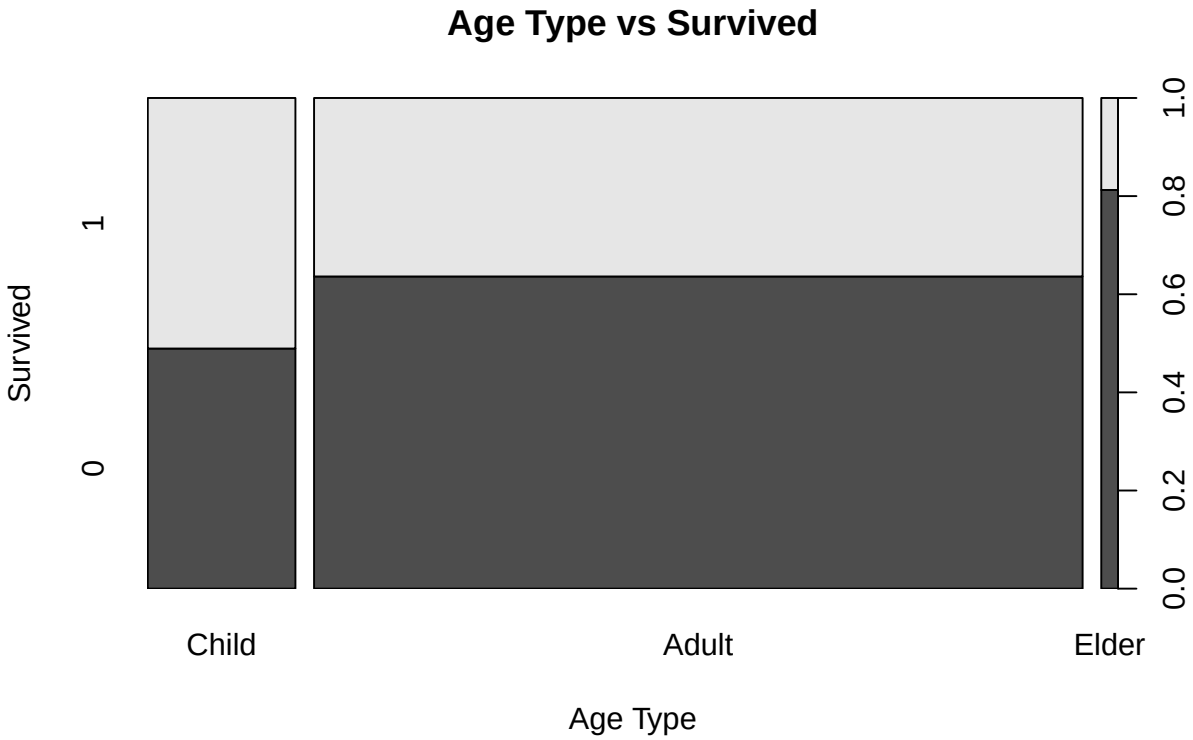
```
titanic$AgeType[titanic$Age < 18] <- "Child"
titanic$AgeType[titanic$Age >= 18 & titanic$Age < 65] <- "Adult"
titanic$AgeType[titanic$Age >= 65] <- "Elder"
titanic$AgeType <- ordered(titanic$AgeType, c("Child", "Adult", "Elder"))
```

```
# Viajeros según el embarque
  local({
   .Table <- with(titanic, table(AgeType ))
   cat("\ncounts:\n")
   print(.Table)
   cat("\npercentages:\n")
   print(round(100*.Table/sum(.Table), 2))
   })
```

```
##
## counts:
## AgeType
## Child Adult Elder
##   189  1100    20
##
```

```
## percentages:
## AgeType
## Child Adult Elder
## 14.44 84.03  1.53
```

```
with(titanic, plot(AgeType,Survived, xlab="Age Type", ylab="Survived", main ="Age Type vs Survived"))
```

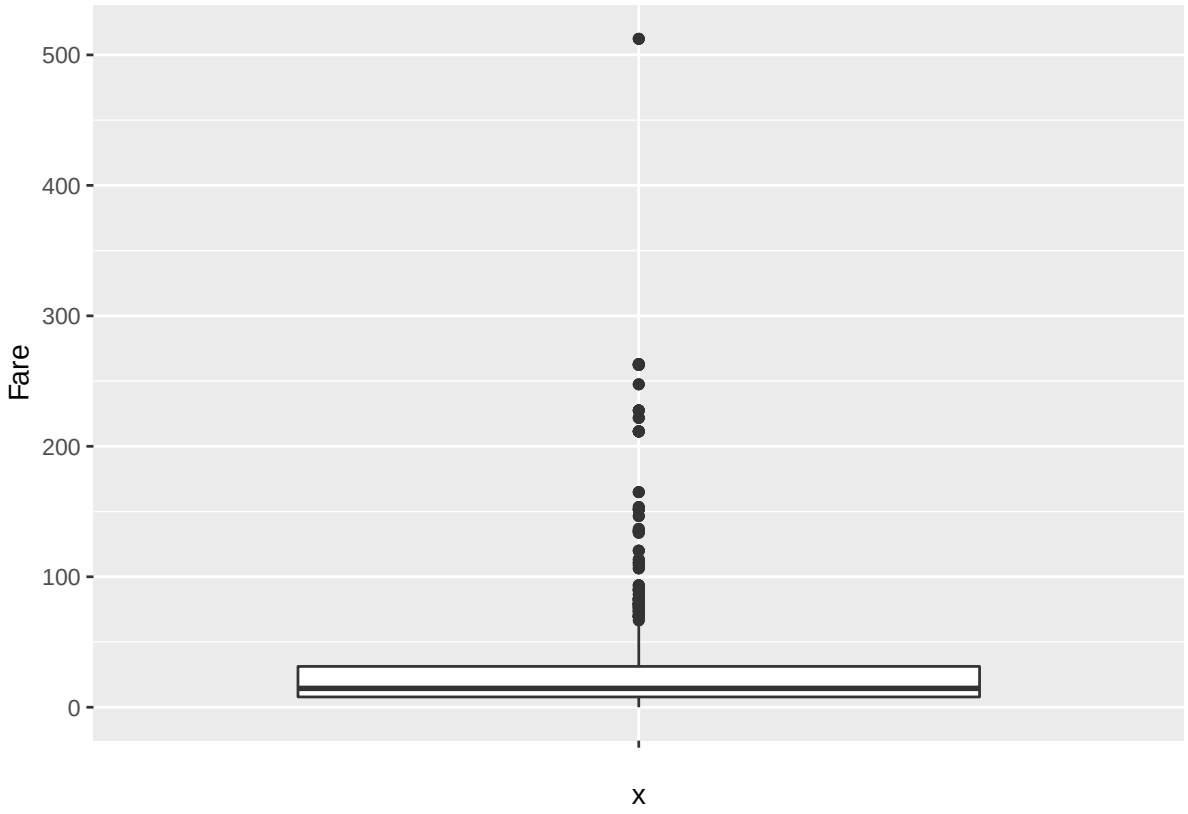



```
#Eliminaos la propiedad Age
properties <- properties[!properties %in% "Age"]
# Añadimos la variable Deck
properties <- c(properties, "AgeType")
```

Como podemos observar en la gráfica los niños también tienen mayor probabilidad de sobrevivir respesto al resto de las clases y los ancinos tiene mucha más probabilidad de no sobrevivir.

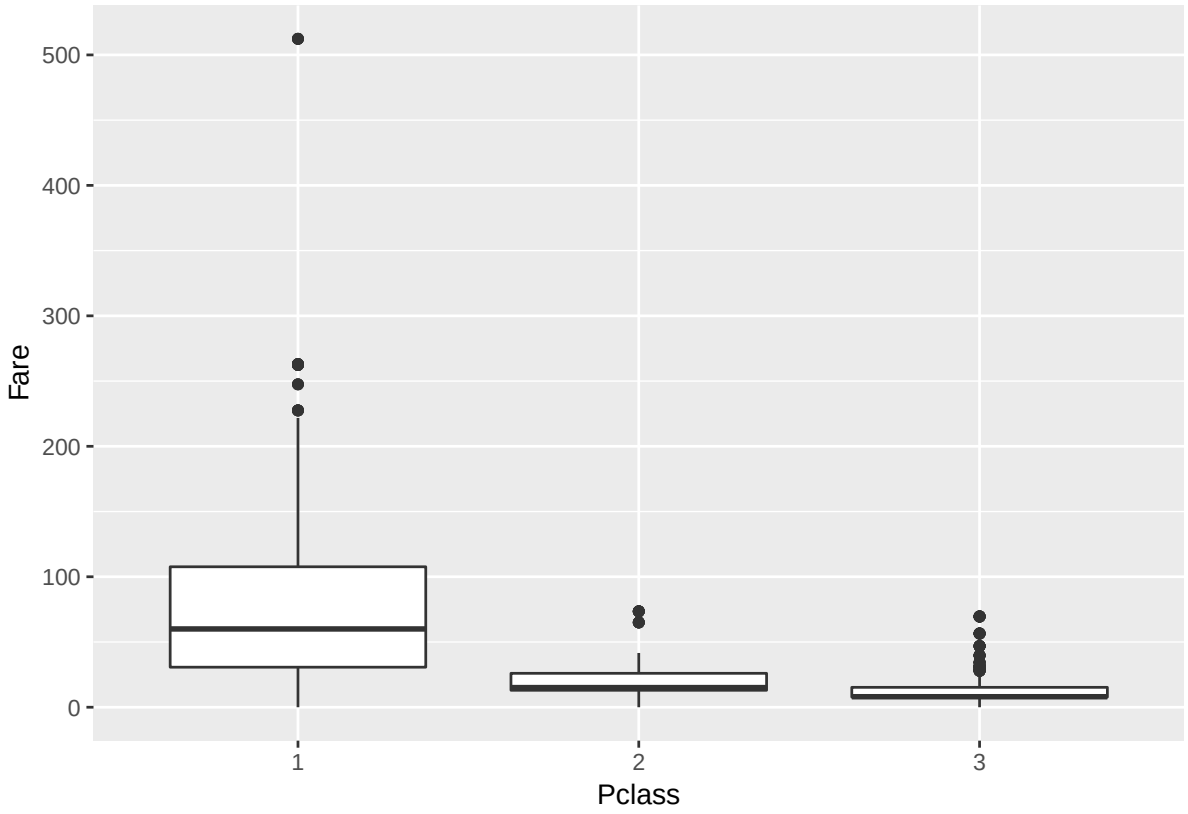
Fare

```
Fare_boxplot <- ggplot(titanic, aes(x="", y=Fare)) +
  geom_boxplot()
Fare_boxplot
```



De las observación de las gráficas, no podemos observar valores extremos que se puedan considerar erroneos. Pero parece razonable que los precios corresponda con el tipo de clase. Por lo que ahora haremos un gráfico de boxplot catalogados por clase.

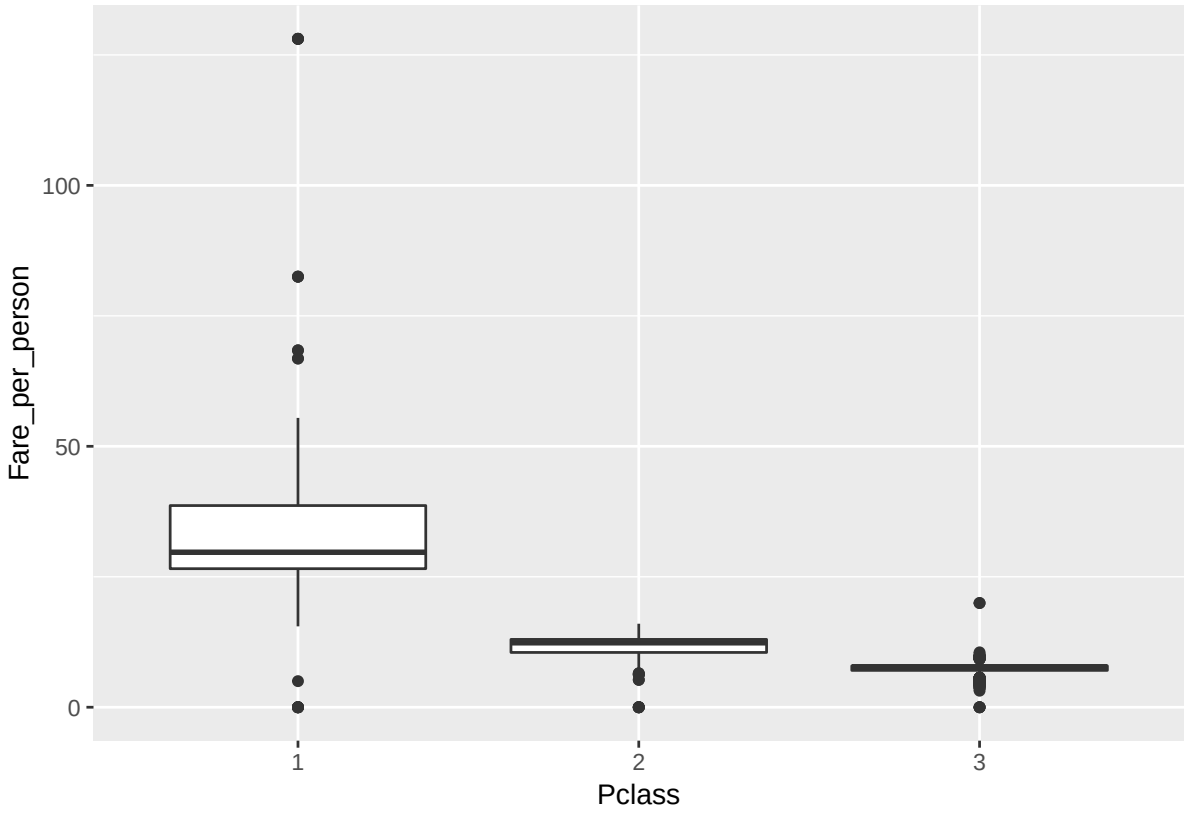
```
Fare_boxplot_Pclass <- ggplot(titanic, aes(x=Pclass, y=Fare)) +
  geom_boxplot()
Fare_boxplot_Pclass
```



Para verificar, los valores extremos podemos calcular el precio por persona y relacionarlo con la clase.

```
# Calculamos el número de pasajeros por ticket
titanic$person_per_ticket <- sapply(titanic$Ticket,
                     function(x) nrow(subset(titanic, Ticket==x)))
# Calculamos el precio por pasajero
titanic$Fare_per_person=titanic$Fare/titanic$person_per_ticket
```

```
# Representams el boxplot del precio por pasajero frente a la clase
Fare_per_person_boxplot_Pclass <- ggplot(titanic, aes(x=Pclass, y=Fare_per_person)) +
  geom_boxplot()
Fare_per_person_boxplot_Pclass
```



Además de esta forma de representar los datos el número de valores extremos se reducen de 171 a 160.

```
Table_FppbPclass <- ggplot_build(Fare_per_person_boxplot_Pclass)$data
sum(sapply(Table_FppbPclass[[1]]$outliers, length))
```

```
## [1] 160
```

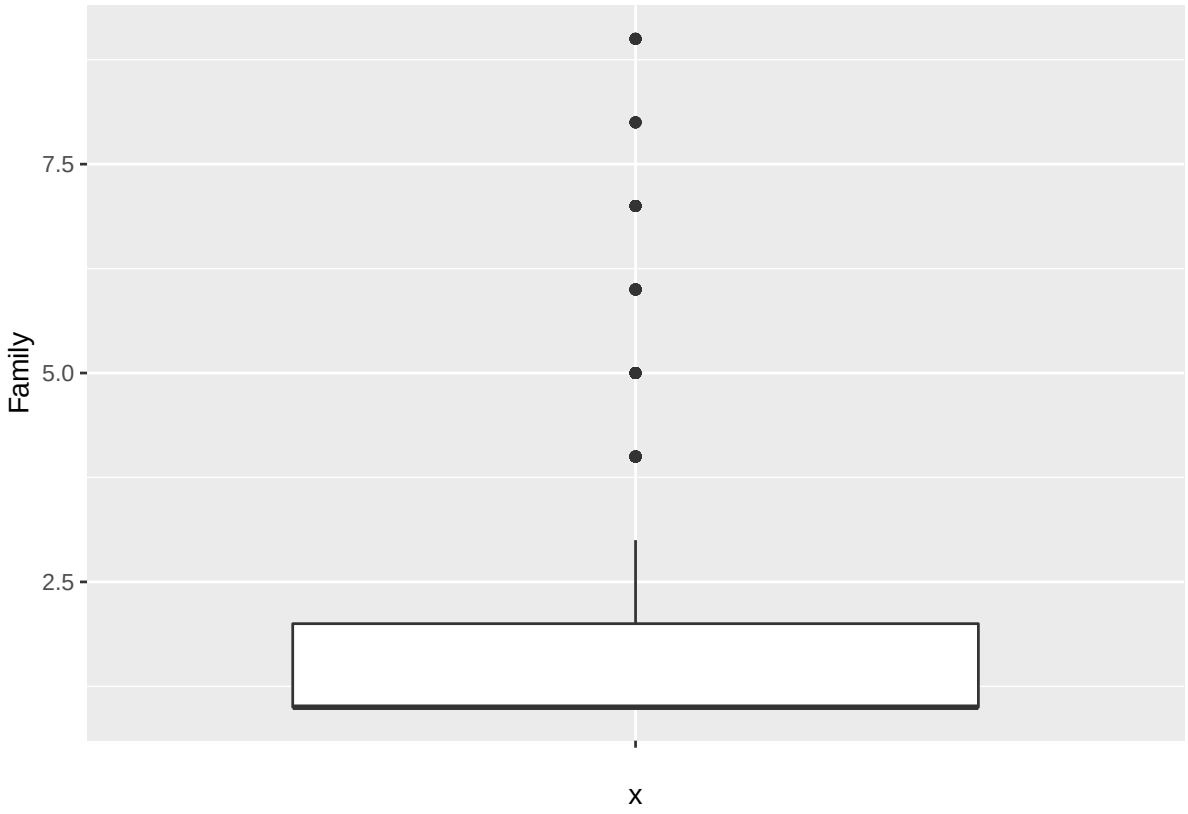
En esta gráfica del precio por persona del billete, observamos una mayor diferencia de los precios por la clase del billete. Aun habiendo precios que son valores extremos, se puede ver que los rangos por cada clase son razonables. Ya que por el precio del billete en el mayoría de los casos se podría determinar a que clase pertenece.

Hay que destacar que en ambas clases hay billetes qeu fueron grátis.

Por lo que parece más razonable utilizar esta variable calculada **Fare_per_person** que la original.

Family

```
Family_boxplot <- ggplot(titanic, aes(x="", y=Family)) +
  geom_boxplot()
Family_boxplot
```



De los datos observamos que hay valore extremos, pero estos valores no parecen erroneos ya que la mayoría de los pasajeros son solteros y hay distintos tipos de familia que van desde los dos miembros hasta los 11 miembros de familia numerosa. Por tanto nuestro campo **FamilyType** calculado también es correcto.

Análisis de los datos.

Selección de los grupos de datos que se quieren analizar/comparar

En nuestro caso, el objetivo es detectar las variables que más contribuyen a explicar si un viajero va a sobrevivir o no, por lo que tendremos que generar un modelo predictivo y/o de clasificación para ubicar a un viajeros según alguna de sus características.

Nuestro dataset ya se nos ha dado en dos conjuntos, uno para estudio y otro para dar solución al problema que se plantea en Kaggle. Estos son los que tenemos un valor en la clase de salidad **Survived**.

Para hacer el estudio, utilizaremos el primer el primer conjunto para entrenamiento y otro para la respuesta final.

Para comparar los modelos realizaremos validaciones cruzadas en los entrenamientos para eviatar sobre ajustes o sub ajustes.

```
# Separamos los datos
train <- titanic %>% filter(not(is.na(Survived)))
rownames(train) <- train$PassengerId

# Datos de  entrega
submission <- titanic %>% filter(is.na(Survived))
rownames(submission) <- submission$PassengerId
```

Comprobación de la normalidad y homogeneidad de la varianza.

Para el estudio de la normalidad y homogeneidad de la varianza en nuestro conjunto utilizaremos el test de normalidad de Anderson-Darling, que básicamente realiza el siguiente contraste de hipótesis:

- H0: No hay diferencias observables entre los datos y la distribución normal
- H1: Existen diferencias observables entre los datos y la distribución normal

```
sapply(train[c(numeric_properties)], ad.test)
```

```
##           Age
## statistic 6.363849
## p.value   1.252479e-15
## method    "Anderson-Darling normality test"
## data.name "X[[i]]"
##           Family
## statistic 117.7682
## p.value   3.7e-24
## method    "Anderson-Darling normality test"
## data.name "X[[i]]"
##           Fare_per_person
## statistic 95.55427
## p.value   3.7e-24
## method    "Anderson-Darling normality test"
## data.name "X[[i]]"
```

Si nos fijamos en los valores de p de todos los atributos y asignamos el valor de significación de 0,05 , observamos que todos ellos son valores muy por debajo del valor de significación. Por tanto no se puede aceptar la hipótesis nula. Por lo que podemos afirmar con un 95 % de fiabilidad que **los datos no siguen una distribución normal**.

Podríamos también aplicar el test de normalidad de Shapiro-Wilk.

```
sapply(train[c(numeric_properties)], shapiro.test)
```

```
##           Age                           Family
## statistic 0.9768755                     0.6360294
## p.value   1.103046e-10                  9.792457e-40
## method    "Shapiro-Wilk normality test" "Shapiro-Wilk normality test"
## data.name "X[[i]]"                      "X[[i]]"
##           Fare_per_person
## statistic 0.6676745
## p.value   1.847716e-38
## method    "Shapiro-Wilk normality test"
## data.name "X[[i]]"
```

Observando igualmente que nuestros valores **no siguen una distribución normal**.

Como ninguna de nuestras variables siguen una distribución normal, para realizar el estudio la homogeneidad de las varianzas, utilizaremos el test de Fligner-Killeen, que compara las varianzas basándose en la mediana. Donde -H0: la varianza es igual entre los grupos -H1: la varianza no es igual entre los grupos

```
train$SurvivedNumeric <- 0
train$SurvivedNumeric[train$Survived ==1]<- 1
test$SurvivedNumeric <- 0
test$SurvivedNumeric[test$Survived ==1]<- 1
fligner.test(Family ~ Survived, data = train)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
```

```
##
## data:  Family by Survived
## Fligner-Killeen:med chi-squared = 19.647, df = 1, p-value =
## 9.317e-06
```

A un 95% de confianza la varianzas **no son iguales** entre los grupos

```
fligner.test(Fare_per_person ~ Survived, data = train)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  Fare_per_person by Survived
## Fligner-Killeen:med chi-squared = 126.45, df = 1, p-value <
## 2.2e-16
```

A un 95% de confianza la varianzas **no son iguales** entre los grupos

```
fligner.test(Age ~ Survived, data = train)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  Age by Survived
## Fligner-Killeen:med chi-squared = 1.443, df = 1, p-value = 0.2297
```

A un 95% de confianza la varianzas **son iguales** entre los grupos.

De las pruebas anteriores, podemos determinar que no podríamos hacer un contraste con ANOVA ya que las poblaciones (distribuciones de probabilidad de la variable dependiente correspondiente a cada factor) no son normales y tampoco cumple la homoscedasticidad.

Aplicación de pruebas estadísticas para comparar los grupos de dato

Para el siguiente estudio nos quedaremos con las propiedades Pclass, Sex, AgeType, Embarked , Title, FamilyType, Deck, Fare_per_person.

De las cuales una es numéricas Fare_per_person y el resto son categóricas

Aunque podríamos utilizar otras dos variables que serían Family o Age si el modelo requiera valores numéricos.

estadístico Chi-cuadrado

Como nuestra clase de salida es de tipo categórica y no numérica, y la mayoría de nuestras variables también lo son podemos hacer una análisis de contraste basado de Chi-cuadrado para ver la dependencia o independencia de dos variables de nuestra muestra.

Nuestra hipótesis serán:

- H0: Las variables son independientes por lo que una variable no varía entre los distintos niveles de la otra variable.
- H1: Las variables son dependientes, una variable varía entre los distintos niveles de la otra variable.

Escogemos un nivel de significación del 0,05.

En primer lugar estudiaremos la dependencia entre la **Pclass** y nuestra variable de salida **Survived**

```
with(titanic, addmargins(table(Pclass, Survived)))
```

```
##       Survived
## Pclass   0   1 Sum
##    1    80 136 216
##    2    97  87 184
##    3   372 119 491
##    Sum 549 342 891
```

```
chisq.test(x =with(titanic, table(Pclass, Survived)))
```

```
##
##  Pearson's Chi-squared test
##
## data:  with(titanic, table(Pclass, Survived))
## X-squared = 102.89, df = 2, p-value < 2.2e-16
```

Como el valor de p-value es menor que 0.05 podemos rechazar la hipótesis nula por lo que podemos decir que la variable **Survived** es dependiente de la varible **Pclass*.

Ahora utilizaremos las varibales **Sex** y nuestra variable de salida **Survived**

```
with(titanic, addmargins(table(Sex, Survived)))
```

```
##         Survived
## Sex        0   1 Sum
##   female  81 233 314
##   male   468 109 577
##   Sum    549 342 891
```

```
chisq.test(x =with(titanic, table(Sex, Survived)))
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  with(titanic, table(Sex, Survived))
## X-squared = 260.72, df = 1, p-value < 2.2e-16
```

Al igual que en el caso anterior pordemos rechazar la hipótesis nula y la supervivencia depende del género.

Si aplicamos el mismo contrates a **Title** frente a **Survived**

```
with(titanic, addmargins(table(Title, Survived)))
```

```
##         Survived
## Title      0   1 Sum
##   Master  17  23  40
##   Miss    55 130 185
##   Mr     451  86 537
##   Mrs     26 103 129
##   Sum    549 342 891
```

```
chisq.test(x =with(titanic, table(Title, Survived)))
```

```
##
##  Pearson's Chi-squared test
##
## data:  with(titanic, table(Title, Survived))
## X-squared = 293.09, df = 3, p-value < 2.2e-16
```

También observamos que la supervivencia depende del título.

Si lo aplicamos al embarked

```
with(titanic, addmargins(table(Embarked, Survived)))
```

```
##          Survived
## Embarked   0   1 Sum
##      C    75  95 170
##      Q    47  30  77
##      S   427 217 644
##      Sum 549 342 891
```

```
chisq.test(x =with(titanic, table(Embarked, Survived)))
```

```
##
##  Pearson's Chi-squared test
##
## data:  with(titanic, table(Embarked, Survived))
## X-squared = 28.005, df = 2, p-value = 8.294e-07
```

Observamos de nuevo la dependencia entre el embarque y la supervivencia

Si lo aplicamos a la edad

```
with(titanic, addmargins(table(AgeType, Survived)))
```

```
##         Survived
## AgeType   0   1 Sum
##   Child  69  72 141
##   Adult 467 267 734
##   Elder  13   3  16
##   Sum   549 342 891
```

En el caso de la **AgeType** con la supervivencia

```
chisq.test(x =with(titanic, table(AgeType, Survived)))
```

```
##
##  Pearson's Chi-squared test
##
## data:  with(titanic, table(AgeType, Survived))
## X-squared = 13.444, df = 2, p-value = 0.001204
```

Vemos la dependencia de la edad también con la supervivencia

En el caso de la **FamilyType** con la supervivencia

```
chisq.test(x =with(titanic, table(FamilyType, Survived)))
```

```
##
##  Pearson's Chi-squared test
##
## data:  with(titanic, table(FamilyType, Survived))
## X-squared = 74.537, df = 2, p-value < 2.2e-16
```

Vemos la dependencia del número de familiares también con la supervivencia

Si lo aplicamos a la Deck

```
with(titanic, addmargins(table(Deck, Survived)))
```

```
##           Survived
## Deck         0   1 Sum
##   A          8   7  15
##   B         12  35  47
##   C         24  35  59
```

```
##   D          8  25  33
##   E          8  24  32
##   F          5   8  13
##   G          2   2   4
##   No Cabin 481 206 687
##   T          1   0   1
##   Sum      549 342 891
```

En el caso de la **Deck** con la supervivencia

```
chisq.test(x =with(titanic, table(Deck, Survived)))
```

```
## Warning in chisq.test(x = with(titanic, table(Deck, Survived))): Chi-
## squared approximation may be incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  with(titanic, table(Deck, Survived))
## X-squared = 99.164, df = 8, p-value < 2.2e-16
```

En este caso vemos que el resultado determina la dependencia entre las dos varibles, pero da un aviso ya que la aproximación es incorrecto.

Esto es debido a que la prueba se basa en los conteos de cada valor distribuyen de manera más o menos normal. Si muchos de los conteos esperados son muy pequeños, la aproximación puede ser deficiente. Como se puede ver en el caso de T Y G donde tenemos 1 ,2 ,0 cuentas para esa valor de la variable Deck respecto a la clase de salida.

Por ello no esta variable **no la tendremos** en cuenta en nuestros modelos.

Modelo de regresión logística

Crearemos un modelo de regresión logística con los predictores anteriores, usando el conjunto de entrenamiento (*train*).

En primer lugar preparamos los datos para utilizar la regresión logísitica con el paquete **caret*

```
train_Logit <- train[c(properties, "Survived")]

# Convertimos las variables para factores que funcione en train
train_Logit$Survived <- make.names(train_Logit$Survived,  unique = FALSE)
train_Logit$Pclass <- make.names(train_Logit$Pclass,  unique = FALSE)
```

Para la comprobación del modelo utilizamos una validación cruzada, intentando evitar la sobre o infra estimación del modelo. Utilizaremos las misma validación cruzada para todos los modelos.

```
# COntrol genérico
trainControl <- trainControl(method="repeatedcv",
                             number=10,
                             repeats=5,
                             p=0.8,
                             search='grid',
                             classProbs = TRUE,
                             savePredictions = TRUE,
                             summaryFunction = twoClassSummary)
```

```
cv_logit <- train(Survived ~ Pclass + Title +  Embarked+ FamilyType + AgeType + Fare_per_person,
                             data= train_Logit,
                             method = "glm",
                             trControl = trainControl,
```

```
                              metric = "ROC",
                              preProc = c("center", "scale"))
```

De los resultados de nuestro modelo, vemos que tanto la Sensibilidad como la Especificidad son bastante buenos y podría ser un buen modelo. Pero lo compararemos posteriormente con el resto de modelos.

```
cv_logit
```

```
## Generalized Linear Model
##
## 891 samples
##   6 predictor
##   2 classes: 'X0', 'X1'
##
## Pre-processing: centered (12), scaled (12)
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 802, 802, 802, 802, 801, 802, ...
## Resampling results:
##
##   ROC        Sens       Spec
##   0.8715741  0.8791178  0.7281176
```

Modelo de clasificación con Random-forest

Como la mayoría de las variables que tenemos las hemos categorizado, una modelo de clasificación que podríamos optar es por un Random-forest.

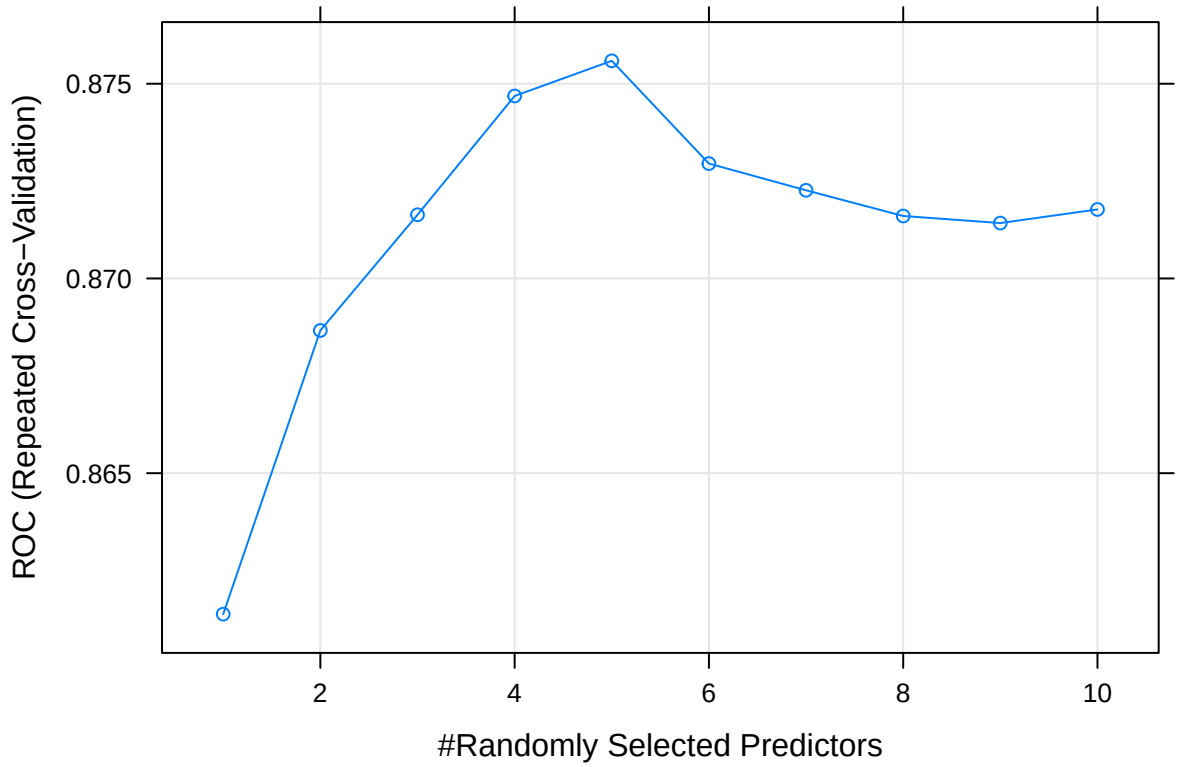
```
train_RF<- train[c(properties, "Survived")]

# Convertimos las variables para factores que funcione en train
train_RF$Survived <- make.names(train_RF$Survived,  unique = FALSE)

train_RF$Pclass <- make.names(train_RF$Pclass,  unique = FALSE)

# Creamos unos rángos para los hiperparámetros

grid_RF <- expand.grid(.mtry=seq(1, 10, by=1))
```

Entrenamos nuestro modelo con la validación cruzada y ponemos como hiperparámetro el valor mtry (número de variable aleatoriamente muestreados como candidatos en cada partición).

```
cv_RF <- train(Survived ~ Pclass + Title +  Embarked + FamilyType + AgeType + Fare_per_person,
                              data= train_RF,
                              method = "rf",
                              trControl = trainControl,
                              metric = "ROC",
                              preProc = c("center", "scale"),
                              tuneGrid=grid_RF)
```

Ahora representamos los valores obtenidos de nuestra busqueda de paramátros mtry

```
plot(cv_RF)
```



Como podemos observar la mejor precisión la obtenemos con el mtry=5. Lo cual prensentando el resumen de nuestro modelo también nos los dice.

```
cv_RF
```

```
## Random Forest
##
## 891 samples
##   6 predictor
##   2 classes: 'X0', 'X1'
##
## Pre-processing: centered (12), scaled (12)
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 802, 802, 802, 801, 802, 802, ...
## Resampling results across tuning parameters:
##
##   mtry  ROC        Sens       Spec
##    1    0.8613807  0.9479192  0.6098655
##    2    0.8686649  0.9231650  0.6864706
##    3    0.8716353  0.9246263  0.6800840
##    4    0.8746858  0.9209764  0.6795462
##    5    0.8755881  0.9060539  0.6865546
##    6    0.8729502  0.8940202  0.6928908
##    7    0.8722641  0.8900202  0.7051261
##    8    0.8716041  0.8878182  0.7040000
##    9    0.8714228  0.8852660  0.7116303
##   10    0.8717723  0.8819933  0.7151429
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 5.
```

De los tantos obtenidos para mtry=5 observamos una mejora del area sobre la curva ROC por lo que parece que este modelo es mejor que la regresión logística.

SVM

Otro modelo que podemos utilizar para predecir la supervivencia de un viajero sería SVM (Support Vector Machine).

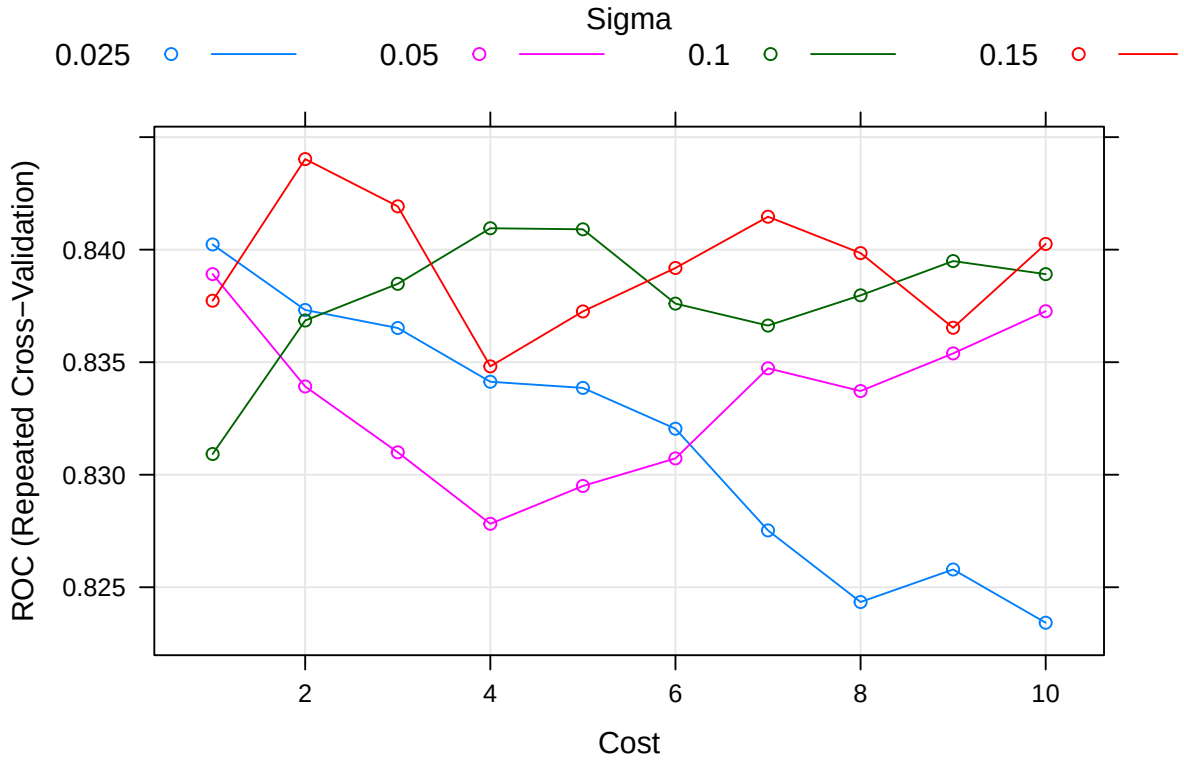
```
train_SVM<- train[c(properties, "Survived")]

# Convertimos las variables para factores que funcione en train
train_SVM$Survived <- make.names(train_SVM$Survived,  unique = FALSE)

train_SVM$Pclass <- make.names(train_SVM$Pclass,  unique = FALSE)

# Creamos unos rángos para los hiperparámetros

grid_SVM <- expand.grid(.sigma=c(0.025, 0.05, 0.1, 0.15), .C=seq(1, 10, by=1))
```

Entrenamos nuestro modelo con la validación cruzada y asignamos el rango de hiperparámetros sigma y coste (c) para buscar el mejor modelo

```
cv_SVM <- train(Survived ~ Pclass + Title +  Embarked+ FamilyType + AgeType + Fare_per_person,
                            data= train_RF,
                            method = "svmRadial",
                            trControl = trainControl,
                            metric = "ROC",
                            preProc = c("center", "scale"),
                            tuneGrid=grid_SVM)
```

Una vez entrenado el modelo representamos los valores de nuestros hiperparámetros para buscar la mejor opción de nuestro model SVM.

```
plot(cv_SVM)
```



Como se puede ve la mejor opción es Sigma = 0.15 y coste=2

```
cv_SVM
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 891 samples
##   6 predictor
##   2 classes: 'X0', 'X1'
##
## Pre-processing: centered (12), scaled (12)
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 802, 802, 802, 802, 801, 801, ...
## Resampling results across tuning parameters:
##
##   sigma  C   ROC        Sens       Spec
##   0.025   1  0.8402314  0.8961549  0.7338655
##   0.025   2  0.8373205  0.8979865  0.7267899
##   0.025   3  0.8365188  0.9110976  0.7062857
##   0.025   4  0.8341306  0.9227542  0.7022185
##   0.025   5  0.8338582  0.9253064  0.6987227
##   0.025   6  0.8320436  0.9263973  0.6993109
##   0.025   7  0.8275254  0.9300337  0.6981345
##   0.025   8  0.8243443  0.9311448  0.6981345
##   0.025   9  0.8257895  0.9311448  0.6969580
##   0.025  10  0.8234233  0.9311448  0.6969580
##   0.050   1  0.8389133  0.9180135  0.7039664
##   0.050   2  0.8339213  0.9289428  0.7004538
##   0.050   3  0.8309958  0.9300337  0.6963697
##   0.050   4  0.8278187  0.9304175  0.6934622
##   0.050   5  0.8295003  0.9300539  0.6934622
##   0.050   6  0.8307257  0.9289630  0.6911092
##   0.050   7  0.8347290  0.9293468  0.6864034
```

```
##   0.050   8  0.8337191  0.9260337  0.6893445
##   0.050   9  0.8353899  0.9264175  0.6881681
##   0.050  10  0.8372642  0.9264175  0.6893445
##   0.100   1  0.8309208  0.9285791  0.6987227
##   0.100   2  0.8368533  0.9271246  0.6928739
##   0.100   3  0.8384818  0.9260337  0.6881681
##   0.100   4  0.8409512  0.9220337  0.6887563
##   0.100   5  0.8409049  0.9209428  0.6875798
##   0.100   6  0.8376028  0.9205791  0.6869916
##   0.100   7  0.8366275  0.9198519  0.6875798
##   0.100   8  0.8379703  0.9194882  0.6869916
##   0.100   9  0.8394929  0.9198519  0.6864034
##   0.100  10  0.8389130  0.9194882  0.6864034
##   0.150   1  0.8377320  0.9260337  0.6952269
##   0.150   2  0.8440247  0.9205791  0.6881681
##   0.150   3  0.8419249  0.9194882  0.6875798
##   0.150   4  0.8348149  0.9198519  0.6887563
##   0.150   5  0.8372550  0.9194949  0.6875798
##   0.150   6  0.8391857  0.9194949  0.6881681
##   0.150   7  0.8414651  0.9169428  0.6864202
##   0.150   8  0.8398480  0.9173131  0.6869748
##   0.150   9  0.8365284  0.9162222  0.6875630
##   0.150  10  0.8402531  0.9151178  0.6875630
##
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.15 and C = 2.
```

El valor del area de la curva ROC es poco peor que el anterior modelo.

Representación de los resultados a partir de tablas y gráficas.

Como hemos visto anteriormente, los modelos parecen lo suficientemente buenos, pero debemos elegir de entre los tres anteriores con el cuál resolveríamos el problema.

Para realizar una comparación entre los métodos representaremos la tabla con los valores de las curvas ROC obtenidas por los tres métodos.

```
columns = c("ROC", "Sens", "Spec", "ROCSD", "SensSD", "SpecSD")

best_result_LOGIT <-  cv_logit$results
best_result_LOGIT$Method <- "Logistic"
best_result_LOGIT$sigma <- NA
best_result_LOGIT$C <- NA
best_result_LOGIT$mtry <- NA

best_result_RF <-  cv_RF$results %>% filter (mtry == cv_RF$bestTune[,"mtry"])
best_result_RF$Method <- "Random-Forest"
best_result_RF$parameter <- NA
best_result_RF$sigma <- NA
best_result_RF$C <- NA

best_result_SVM <- cv_SVM$results %>% filter (C == cv_SVM$bestTune[,"C"] & sigma == cv_SVM$bestTune[,
best_result_SVM$Method <- "SVM"
best_result_SVM$parameter <- NA
best_result_SVM$mtry <- NA
```

```
results <- rbind(best_result_LOGIT, best_result_RF, best_result_SVM)
columns = c("Method", "ROC", "Sens", "Spec", "ROCSD", "SensSD", "SpecSD")
results[columns]
```

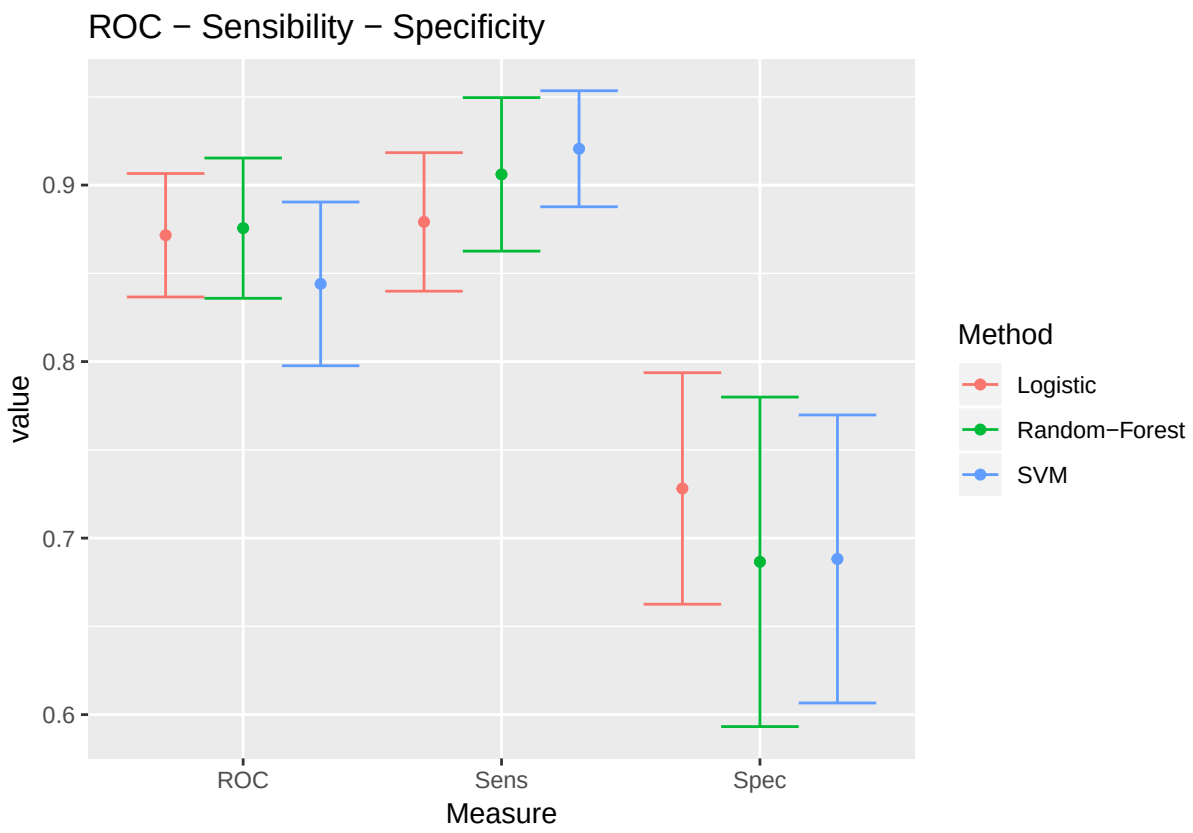
```
##          Method       ROC      Sens      Spec      ROCSD     SensSD
## 1      Logistic 0.8715741 0.8791178 0.7281176 0.03497399 0.03923357
## 2 Random-Forest 0.8755881 0.9060539 0.6865546 0.03973054 0.04348912
## 3           SVM 0.8440247 0.9205791 0.6881681 0.04637909 0.03286262
##       SpecSD
## 1 0.06558473
## 2 0.09334025
## 3 0.08158318
```

De la tabla podemos observar que el peor modelo sería el **SVM** según el área de la curva ROC. Pero es el que mejor Sensibilidad tiene.

Para tener una mejor visión, podemos representar el valor media y con su desviación típica

```
table_roc <- results[c("Method", "ROC", "ROCSD")]
table_roc$Measure <- "ROC"
table_roc <- table_roc %>% rename (
    mean = ROC,
    sd = ROCSD
    )

table_sens <- results[c("Method", "Sens", "SensSD")]
table_sens$Measure <- "Sens"
table_sens <- table_sens %>% rename(
     mean = Sens,
    sd = SensSD
    )

table_spec <- results[c("Method", "Spec", "SpecSD")]
table_spec$Measure <- "Spec"
table_spec <- table_spec %>% rename(
    mean = Spec,
    sd = SpecSD
    )


compare_plot <- rbind(table_roc, table_sens, table_spec)
```

```
#compare_plot$Method <- cbind( results["Method", "ROC", "ROCSD"])

ggplot(compare_plot, aes(x = Measure, colour=Method)) +
  geom_errorbar(aes(ymax = mean + sd, ymin = mean - sd),
                position = "dodge") +
  geom_point(position=position_dodge(width=0.9), aes(y=mean, colour=Method))  +
  ggtitle("ROC - Sensibility - Specificity") +
  labs (y ="value")
```



Con esta gráfica vemos cual es el mejor método según las tres medidas:

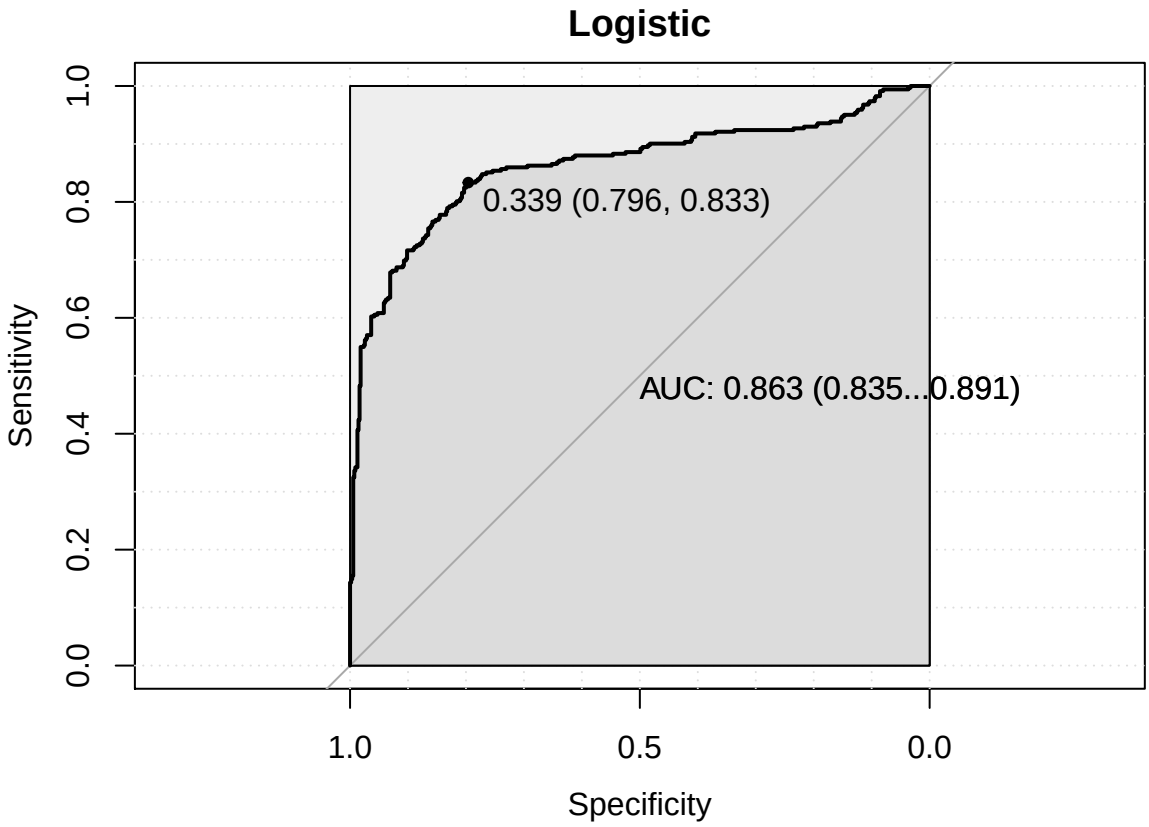
- Según la curva ROC: Una medida más general del modelo de predecir positivos y negativos correctamente el mejor es **Random-Forest**
- Segun la sensibilidad: Es decir una mayor predisposición para catalogar como sobrevivientes entonces el mejor modelo es **SVM**
- Según la especifidad: La mayor predisposición del modelo para catalogar como no supervivientes entonces el mejor modelos es **Regresión Logísitica**

Para verificar este resultado podemos dibujar las gráficas del ROC de cada uno de los modelos (la media de las validaciones cruzadas)

```
roc_logit = roc(as.numeric(cv_logit$trainingData$.outcome=='X1'),aggregate(X1~rowIndex,cv_logit$pred,
            smoothed = TRUE,
            # arguments for ci
            ci=TRUE, ci.alpha=0.95, stratified=FALSE,
            # arguments for plot
            plot=TRUE,
            auc.polygon=TRUE,
            max.auc.polygon=TRUE,
            grid=TRUE,
            print.auc=TRUE,
            show.thres=TRUE,
            print.thres="best",
            main="Logistic"
            )
```

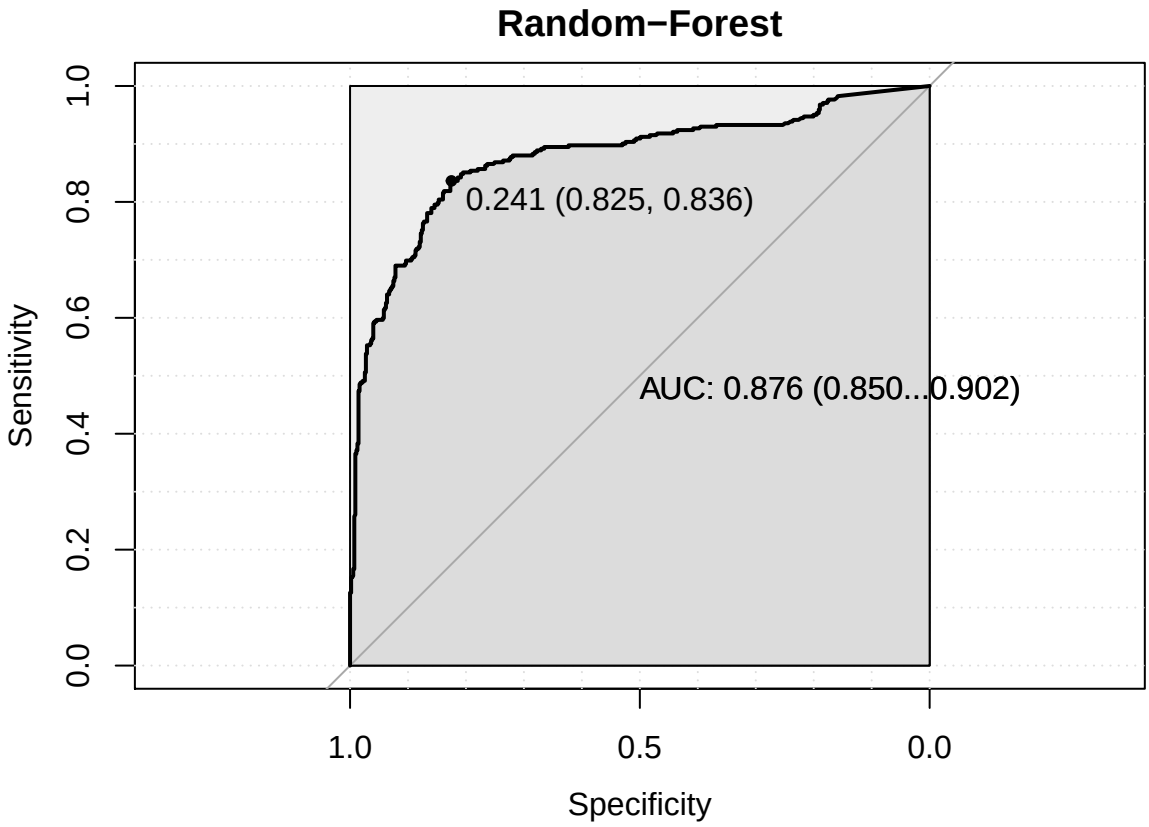
```
## Setting levels: control = 0, case = 1
```

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




```
roc_RF = roc(as.numeric(cv_RF$trainingData$.outcome=='X1'),aggregate(X1~rowIndex,cv_RF$pred,mean)[,'X
             smoothed = TRUE,
             # arguments for ci
             ci=TRUE, ci.alpha=0.9, stratified=FALSE,
             # arguments for plot
             plot=TRUE,
             auc.polygon=TRUE,
             max.auc.polygon=TRUE,
             grid=TRUE,
             print.auc=TRUE,
             show.thres=TRUE,
             print.thres="best",
             main="Random-Forest"
             )
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```



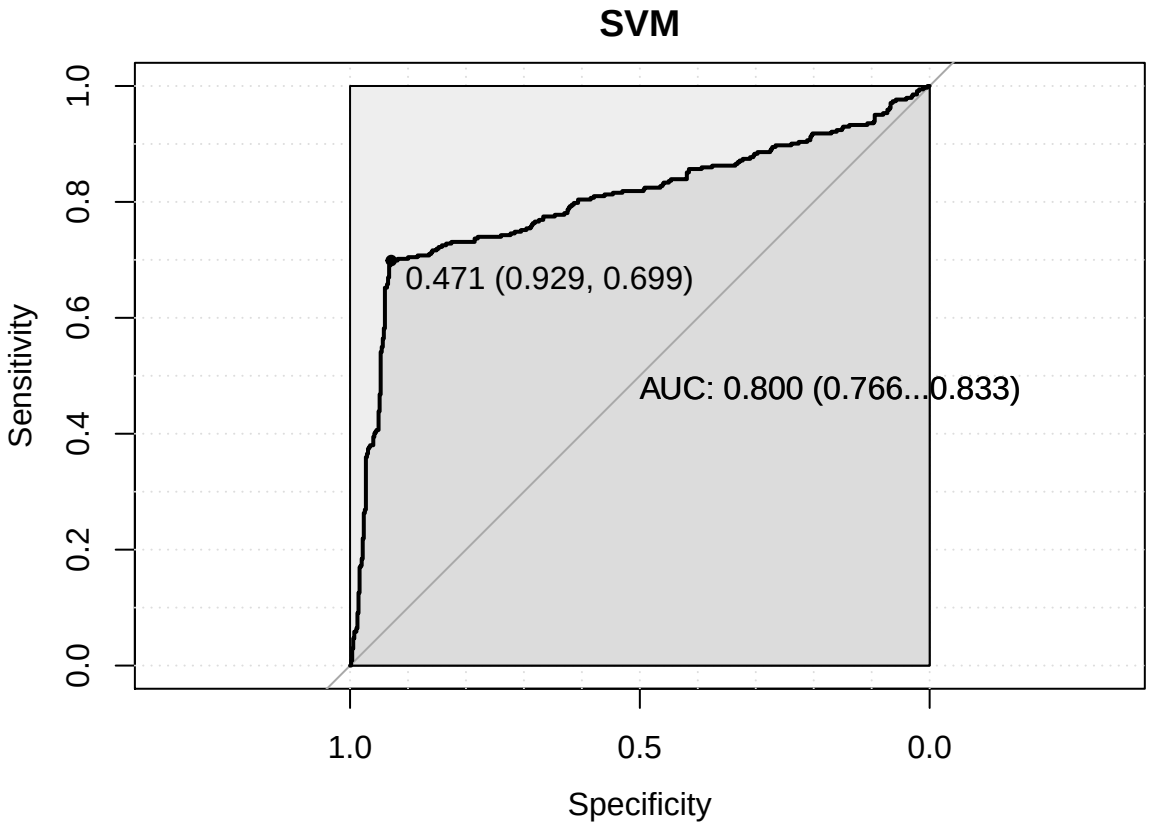

```
# Obtenemos la media de la curva ROC de las distintas validaciones
best_model_SVM <-  cv_SVM$pred %>% filter (C == cv_SVM$bestTune[,"C"] & sigma == cv_SVM$bestTune[,"si
 cv_SVM$results %>% filter (C == cv_SVM$bestTune[,"C"] & sigma == cv_SVM$bestTune[,"sigma"])
```

```
##   sigma C       ROC      Sens      Spec      ROCSD     SensSD     SpecSD
## 1  0.15 2 0.8440247 0.9205791 0.6881681 0.04637909 0.03286262 0.08158318
```

```
roc_SVM = roc(as.numeric(cv_SVM$trainingData$.outcome=='X1'),aggregate(X1~rowIndex,best_model_SVM,mea
            smoothed = TRUE,
            # arguments for ci
            ci=TRUE, ci.alpha=0.9, stratified=FALSE,
            # arguments for plot
            plot=TRUE,
            auc.polygon=TRUE,
            max.auc.polygon=TRUE,
            grid=TRUE,
            print.auc=TRUE,
            show.thres=TRUE,
            print.thres="best",
            main = "SVM"
            )
```

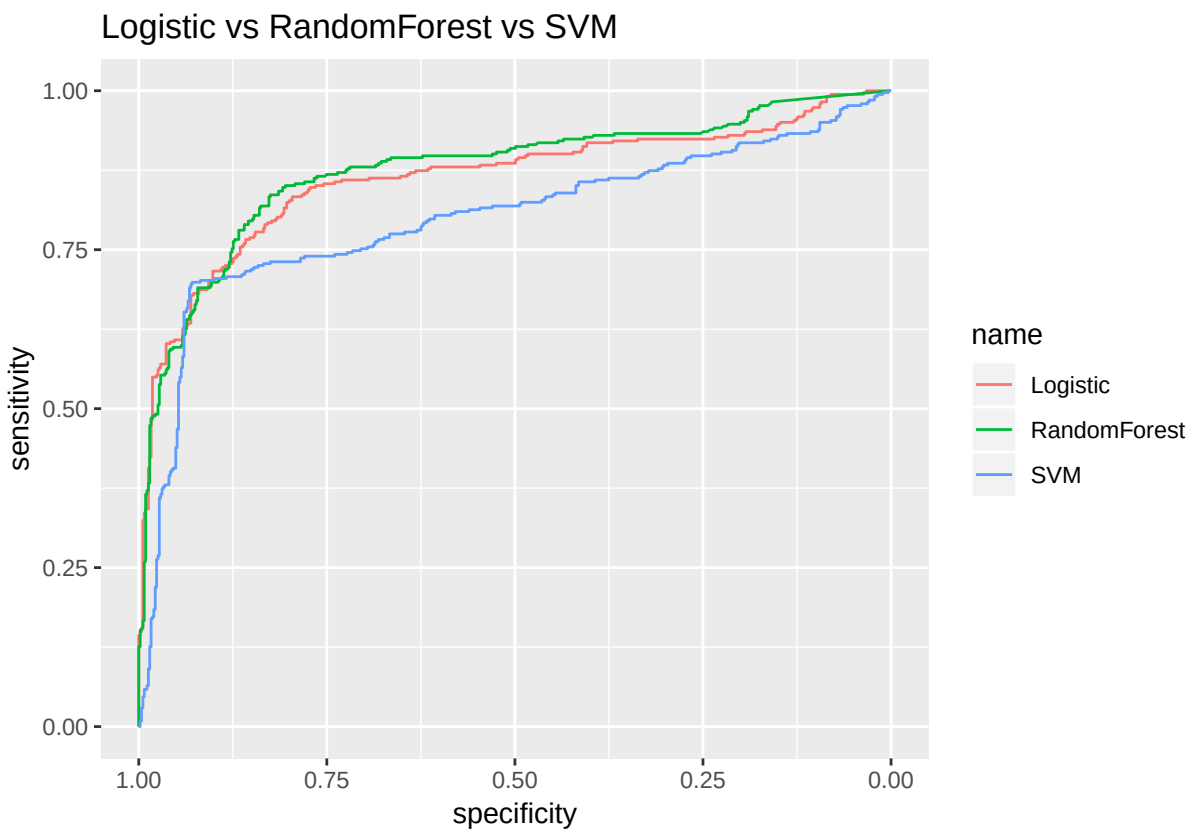
```
## Setting levels: control = 0, case = 1
```

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



Si enfrentamos todos los valores en una mísma gráfica podemos observar que el mejor modelo como norma general sería **Random-Forest**.

```
g <-ggroc(list(Logistic = roc_logit, RandomForest = roc_RF, SVM= roc_SVM)) +
  ggtitle("Logistic vs RandomForest vs SVM")

g
```



El modelo **logístico** en esta grafica vemos que es muy similar al **Random-Forest**, pero es algo peor tanto con esta gráfica como con los valores medios de nuestras medidas de ROC, Sensibilidad y Especificidad.

Tambiéns se puede observar que si queremos que nuestro modelo catalogáse mejor los casos de supervivencia (Sensibilidad), hay un corte del umbral donde el modelo de *SVM* es mucho mejor, como también vimos chequando las medias de la medias. Con esta gráfica se ve en la esquina azul que además es el mejor umbral que obteníamos en el modelo *SVM*. Este modelo podría ser mejor con dicho umbral en el caso que quisieramos catalogar mejor los supervivientes, pero no nos importante catalogar con mayor porcentajes de error los que no sobreviven.

Resolución del problema

Como en nuestro caso, queremos catalogar correctamente el mayor número posible, optaríamos por la opción del modelo de *RandomForest* y utilizaríamos el mejor umbral detectado.

```
# Seleccionamos el mejor umbral
best_threshold <- coords(roc_RF, "best", ret = "threshold", transpose=TRUE)

# Preparamos los datos para la predicción

submission_RF<- submission[c(properties, "Survived")]

# Convertimos las variables para factores que funcione en train
submission_RF$Survived <- make.names(submission_RF$Survived,  unique = FALSE)

submission_RF$Pclass <- make.names(submission_RF$Pclass,  unique = FALSE)

output_prob <- predict(cv_RF, newdata = submission_RF ,  type = "prob")
```

```
# Asignamos los resultado con el mejor umbral.

submission_RF<- cbind(submission_RF, output_prob)
submission_RF$Survived <- 0
submission_RF$Survived[submission_RF$X1 >= best_threshold]<- 1
submission_RF$PassengerId <- as.numeric(rownames(submission_RF))
head(submission_RF[c("PassengerId", "Survived")])
```

```
##     PassengerId Survived
## 892         892        0
## 893         893        1
## 894         894        0
## 895         895        0
## 896         896        1
## 897         897        0
```

```
write.csv(submission_RF[c("PassengerId", "Survived")],
          file = "kaggle/output_submission.csv",
          row.names = FALSE)


# Salida con las varriables utilizada en los modelos y el resultado
train$type <- "train"
submission_RF$type <- "result"

output <- rbind(train[c("Survived", "Pclass", "Title",  "Embarked", "FamilyType", "AgeType", "Fare_pe
                submission_RF[c("Survived", "Pclass", "Title",  "Embarked", "FamilyType", "AgeType",
output$PassengerId <- as.numeric(rownames(output))

write.csv(output, file= "kaggle/output.csv")
```

Código

El código se encuentra disponible en

https://github.com/tanakafer/titanic

Dataset

El dataset se puede conseguir en https://www.kaggle.com/c/titanic/data

Contribuciones

Contribuciones	Firma
Investigación previa	FRL
Redacción de las respuestas	FRL
Desarrollo código	FRL

References

Anomymous. n.d. "Large families not good for Survival | Kaggle." Accessed May 13, 2019. https://www.kaggle.com/jasonm/large-families-not-good-for-survival.

Anonymous. n.d. "Basic Feature Engineering with the Titanic Data « triangleinequality." Accessed May 13, 2019. https://triangleinequality.wordpress.com/2013/09/08/basic-feature-engineering-with-the-titanic-data/.

Megan L. Risdal. 2016. "Exploring Survival on the Titanic | Kaggle." https://www.kaggle.com/mrisdal/exploring-survival-on-the-titanic.

Osborne, Jason W. 2010. "Data Cleaning Basics: Best Practices in Dealing with Extreme Scores." *Newborn and Infant Nursing Reviews* 10 (1): 37–43. https://doi.org/10.1053/j.nainr.2009.12.009.