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 se han dividido 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 conunto de test** usado para comprobar lo bien que predice el modelo. En este grupo no se aporta la clase de salida. Sino que este grupo es utilizado para verificar los bien que modelo predice si un pasajero habría sobrevivido o no dependiendo de sus propiedades.

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 tes 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 númeor 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 le asignamos el valor TBD. 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.

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)
# 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.

```
# Echamos un vistazo a los datos
#kable(head(titanic),digits = 3, padding = 2, align = 'r')
head(titanic)
```

```
##   PassengerId Survived Pclass
## 1           1        0      3
## 2           2        1      1
## 3           3        1      3
## 4           4        1      1
## 5           5        0      3
## 6           6        0      3
##                                                 Name    Sex Age SibSp
## 1                             Braund, Mr. Owen Harris   male  22     1
## 2 Cumings, Mrs. John Bradley (Florence Briggs Thayer) female  38     1
## 3                              Heikkinen, Miss. Laina female  26     0
## 4        Futrelle, Mrs. Jacques Heath (Lily May Peel) female  35     1
## 5                            Allen, Mr. William Henry   male  35     0
## 6                                    Moran, Mr. James   male  NA     0
##   Parch           Ticket    Fare Cabin Embarked
## 1     0        A/5 21171  7.2500              S
## 2     0         PC 17599 71.2833   C85        C
## 3     0 STON/O2. 3101282  7.9250              S
## 4     0           113803 53.1000  C123        S
## 5     0           373450  8.0500              S
## 6     0           330877  8.4583              Q
```

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.

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.

```
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" ...
```

```
#Comprobamos  que los PassengerID son los mismos en el dataframe titanic con Survived a NA y los de t
print("____________________")
```

```
## [1] "____________________"
```

```
print( "Diferencias")
```

```
## [1] "Diferencias"
```

```
print("____________________")
```

```
## [1] "____________________"
```

```
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. Remplazamos el valor NA por TBD para crear la variable tipo factor

```
titanic$Survived[is.na(titanic$Survived)] <- "TBD"
titanic$Survived <- as.factor(titanic$Survived)
levels(titanic$Survived)
```

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

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"]
```

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)
# Comprobamos los valores de la clase
levels(titanic$Pclass)
```

```
## [1] "1" "2" "3"
```

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)
```

Del propio campo **Name** podemos obtener el apellido para determinar la familia, el cual podría ser un atrituto útil para los posibles modelos.

```
titanic$Surname <- sapply(titanic$Name,
                      function(x) strsplit(x, split = '[,.]')[[1]][1])
```

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"]
```

El campo **Sex**(género) podría ser útil para nuestros modelos por lo que lo mantenemos. 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. Los dos siguientes atributos **Sibsp**(hermanos, pareja) y **Parch** (padres e hijos) pueden ser interesantes para nuestros modelos, pero también 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")
```

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   TBD: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
##    Surname              Family
##  Length:1309        Min.   : 1.000
##  Class :character   1st Qu.: 1.000
##  Mode  :character   Median : 1.000
##                     Mean   : 1.884
##                     3rd Qu.: 2.000
##                     Max.   :11.000
##
```

```
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 elimnamos de nuestro dataframe

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

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 aparatado.

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)
# 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   TBD: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                 :1014   Length:1309        Master: 61
##  1st Qu.:  7.896   C23 C25 C27   :   6   Class :character   Miss  :264
##  Median : 14.454   B57 B59 B63 B66:   5   Mode  :character   Mr    :782
##  Mean   : 33.295   G6            :   5                      Mrs   :202
##  3rd Qu.: 31.275   B96 B98       :   4
##  Max.   :512.329   C22 C26       :   4
##  NA's   :1         (Other)       : 271
##    Surname             Family
##  Length:1309        Min.   : 1.000
##  Class :character   1st Qu.: 1.000
##  Mode  :character   Median : 1.000
##                     Mean   : 1.884
##                     3rd Qu.: 2.000
##                     Max.   :11.000
##
```

```
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
```

En el resumen vemos que hay 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 y lo eliminamos del dataframe.

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

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)
# Hacemos un sumary
summary(titanic %>% select(properties))
```

```
##  Pclass      Sex           Age            SibSp           Parch
##  1:323   female:466   Min.   : 0.17   Min.   :0.0000   Min.   :0.000
##  2:277   male  :843   1st Qu.:21.00   1st Qu.:0.0000   1st Qu.:0.000
##  3:709                Median :28.00   Median :0.0000   Median :0.000
##                       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          Embarked    Title         Family
##  Min.   :  0.000    :  2    Master: 61   Min.   : 1.000
##  1st Qu.:  7.896   C:270    Miss  :264   1st Qu.: 1.000
##  Median : 14.454   Q:123    Mr    :782   Median : 1.000
##  Mean   : 33.295   S:914    Mrs   :202   Mean   : 1.884
##  3rd Qu.: 31.275                         3rd Qu.: 2.000
##  Max.   :512.329                         Max.   :11.000
##  NA's   :1
```

```
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.

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*

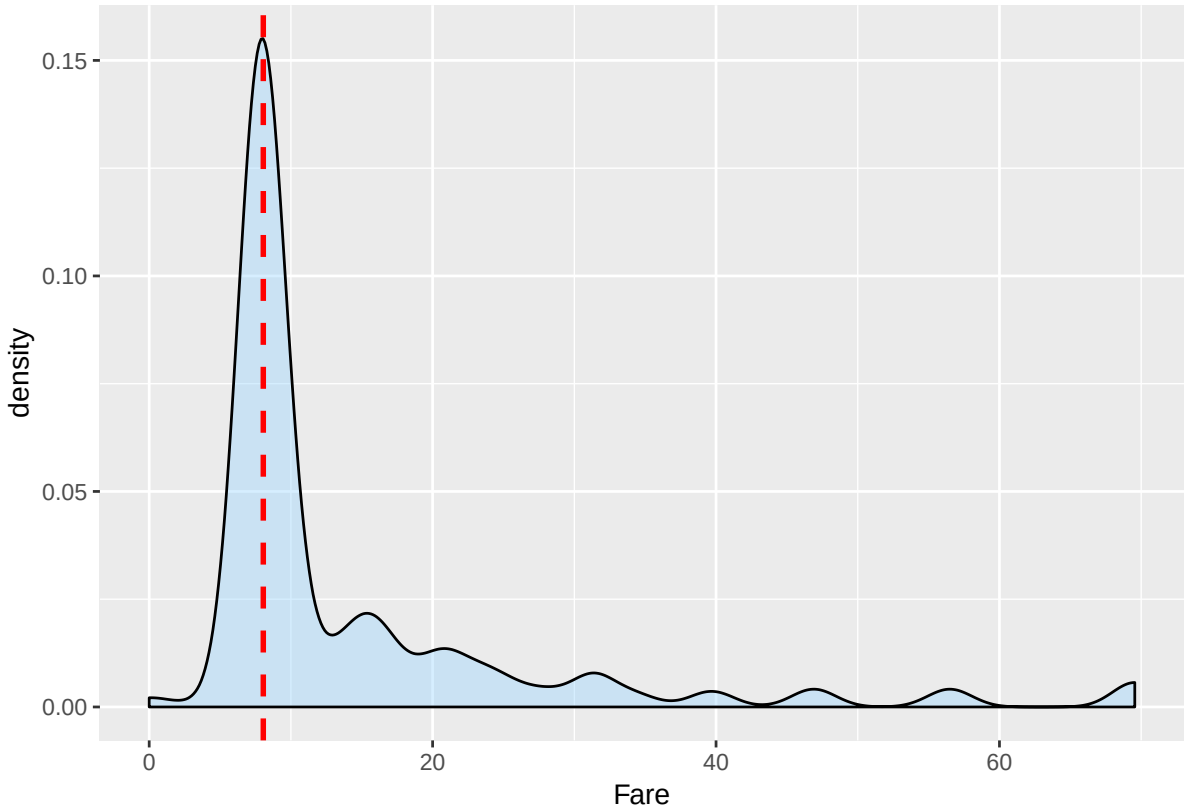
```
titanic %>% filter(is.na(titanic$Fare))
```

```
##   PassengerId Survived Pclass               Name  Sex  Age SibSp Parch
## 1        1044      TBD      3 Storey, Mr. Thomas male 60.5     0     0
##   Ticket Fare Cabin Embarked Title Surname Family
## 1   3701   NA              S    Mr  Storey      1
```

De este pasajero observamos que su embarque fué en *Southampton* ('S') y es de tercera clase.

```
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 visaulización 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"
```

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          SibSp           Parch
##  1: 39   female: 78   Min.   : NA   Min.   :0.0000   Min.   :0.0000
##  2: 16   male  :185   1st Qu.: NA   1st Qu.:0.0000   1st Qu.:0.0000
##  3:208                Median : NA   Median :0.0000   Median :0.0000
##                       Mean   :NaN   Mean   :0.4829   Mean   :0.2433
##                       3rd Qu.: NA   3rd Qu.:0.0000   3rd Qu.:0.0000
##                       Max.   : NA   Max.   :8.0000   Max.   :9.0000
##                       NA's   :263
##       Fare         Embarked    Title         Family
##  Min.   :  0.00    :  0    Master:  8   Min.   : 1.000
```

```
##  1st Qu.:  7.75   C: 58    Miss  : 51   1st Qu.: 1.000
##  Median :  8.05   Q: 73    Mr    :177   Median : 1.000
##  Mean   : 19.82   S:132    Mrs   : 27   Mean   : 1.726
##  3rd Qu.: 22.80                         3rd Qu.: 1.000
##  Max.   :227.53                         Max.   :11.000
##
```

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.

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

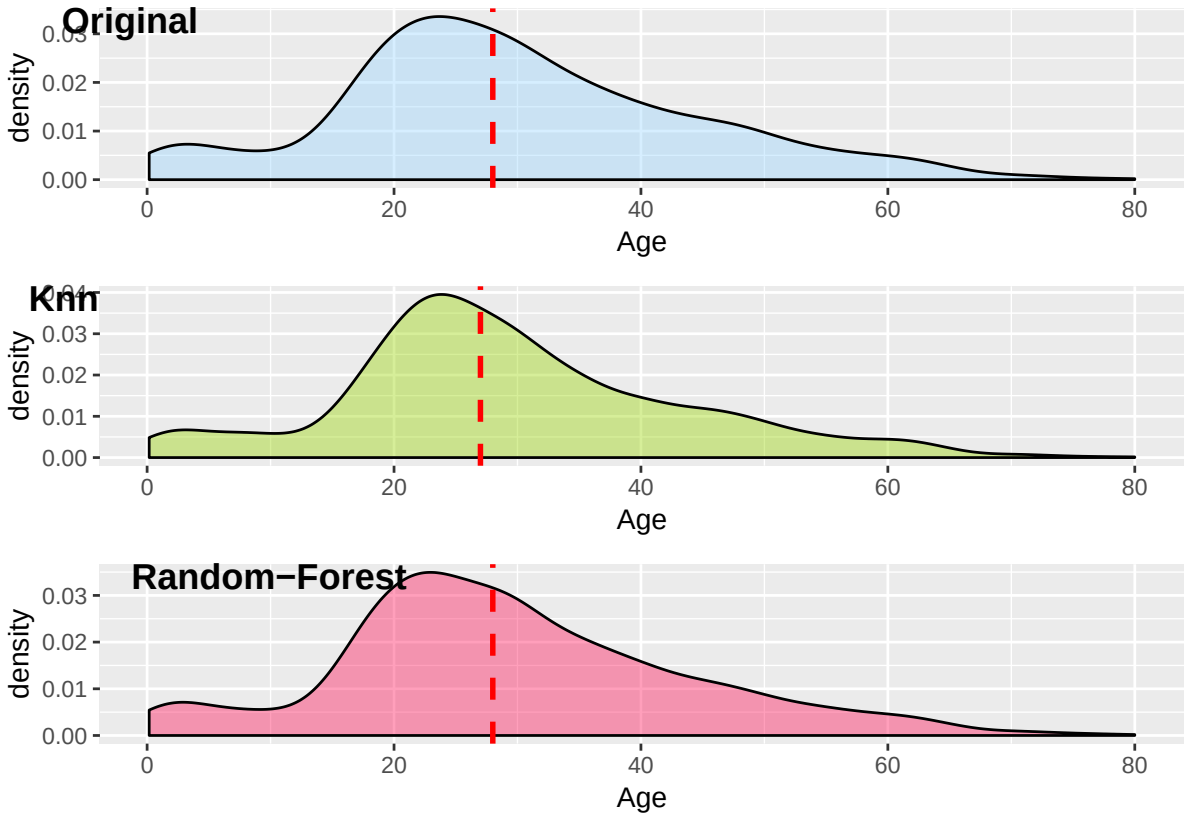
```
##
##  iter imp variable
##   1   1  Age
##   1   2  Age
##   1   3  Age
##   1   4  Age
##   1   5  Age
##   2   1  Age
##   2   2  Age
##   2   3  Age
##   2   4  Age
##   2   5  Age
##   3   1  Age
##   3   2  Age
##   3   3  Age
##   3   4  Age
##   3   5  Age
##   4   1  Age
##   4   2  Age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

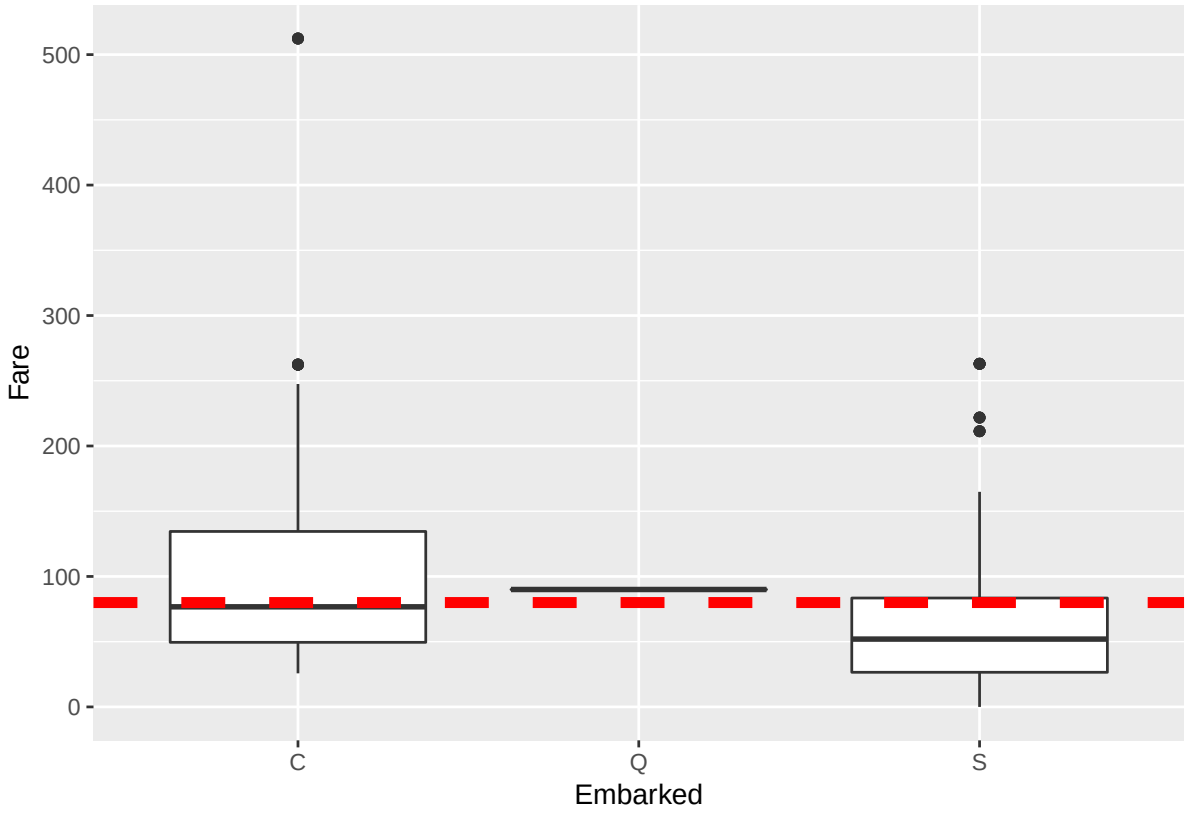
Presentamos los valores con embarque vacio

```
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 Surname Family
## 1 female  38     0     0 113572   80   B28           Miss   Icard      1
## 2 female  62     0     0 113572   80   B28            Mrs   Stone      1
```

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)
```



Identificación y tratamiento de valores externos

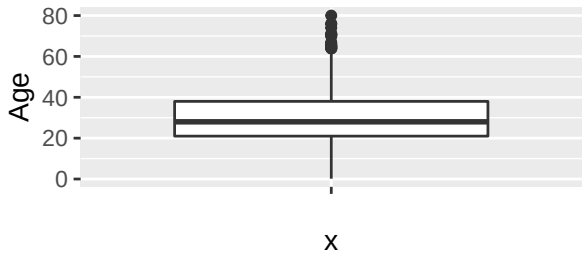
Para detectar la presencia de valores atípicos examinaremos primero el resumen de los cinco números de Tukey, donde podremos observar un análisis descriptivo de los datos De este resumen detectaremos si la media y la mediana están muy separadas, para analizar dichas variables más en detalle posteriormente. Para obtener los datos sólo utilizaremos las variables numéricas **Age**, **SibSp**, **Parch**, **Fare** y la calculada **Family**.

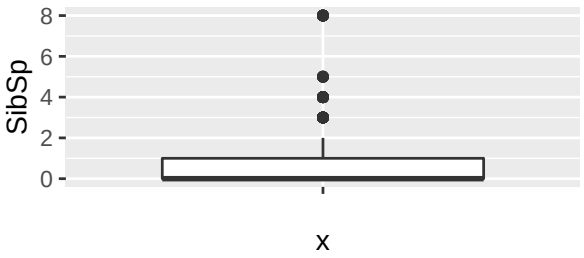
```
numeric_properties <- c ("Age", "SibSp", "Parch", "Fare", "Family")
summary(titanic %>% select(numeric_properties))
```

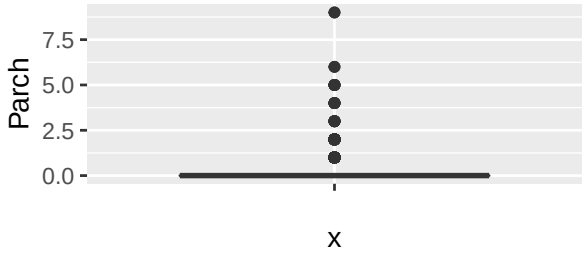
```
##       Age            SibSp            Parch            Fare
##  Min.   : 0.17   Min.   :0.0000   Min.   :0.000   Min.   :  0.000
##  1st Qu.:21.00   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:  7.896
##  Median :28.00   Median :0.0000   Median :0.000   Median : 14.454
##  Mean   :29.61   Mean   :0.4989   Mean   :0.385   Mean   : 33.276
##  3rd Qu.:38.00   3rd Qu.:1.0000   3rd Qu.:0.000   3rd Qu.: 31.275
##  Max.   :80.00   Max.   :8.0000   Max.   :9.000   Max.   :512.329
##      Family
##  Min.   : 1.000
##  1st Qu.: 1.000
##  Median : 1.000
##  Mean   : 1.884
##  3rd Qu.: 2.000
##  Max.   :11.000
```

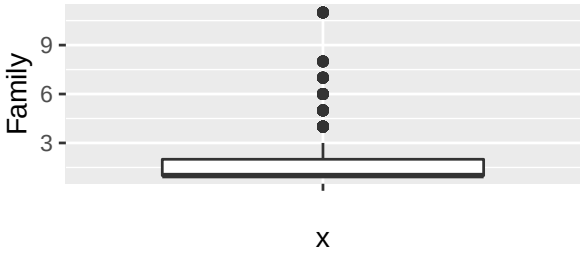
Los cinco números también se representan gráficamente con **boxplot**

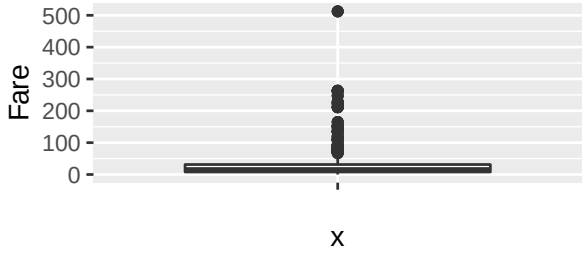
```
Age_boxplot <- ggplot(titanic, aes(x="", y=Age) ) +
  geom_boxplot()
SibSp_boxplot <- ggplot(titanic, aes(x="", y=SibSp)) +
  geom_boxplot()
Parch_boxplot <- ggplot(titanic, aes(x="", y=Parch)) +
  geom_boxplot()
Family_boxplot <- ggplot(titanic, aes(x="", y=Family)) +
  geom_boxplot()
Fare_boxplot <- ggplot(titanic, aes(x="", y=Fare)) +
  geom_boxplot()

boxplots <- ggarrange(Age_boxplot, SibSp_boxplot, Parch_boxplot, Family_boxplot, Fare_boxplot,

                    ncol = 2, nrow =3)
boxplots
```



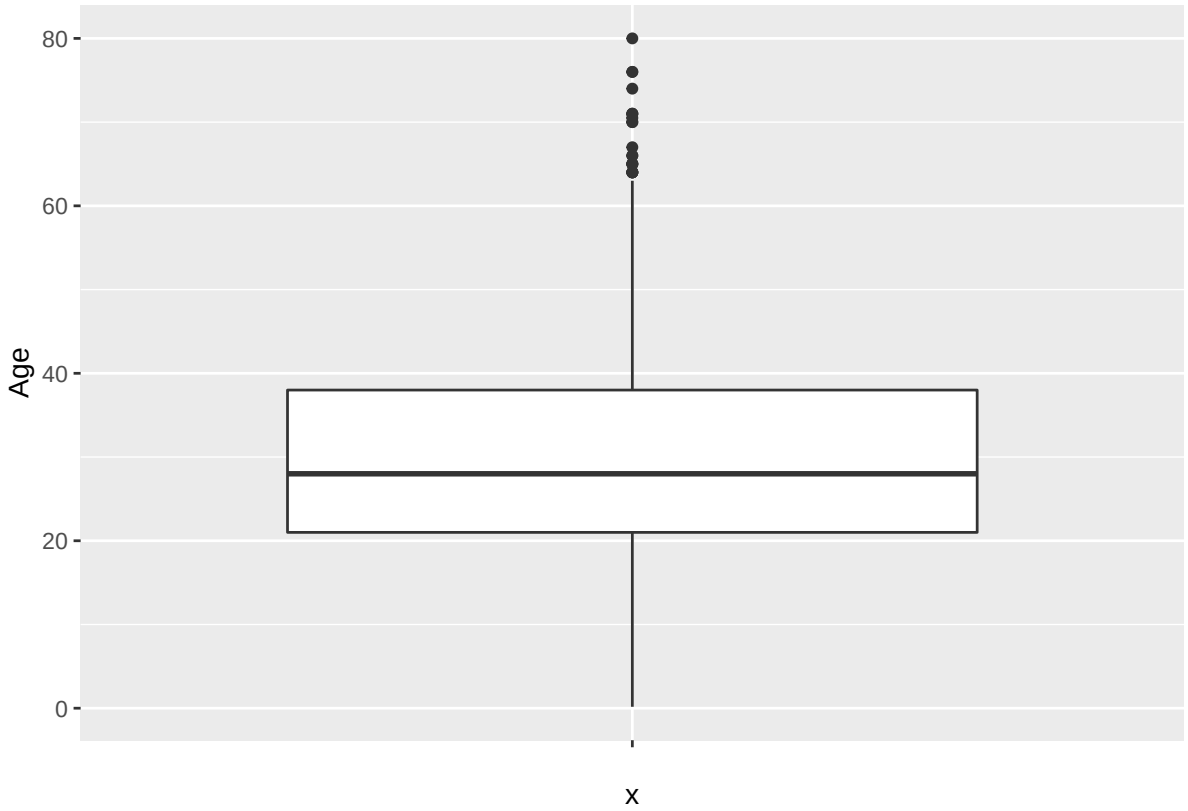








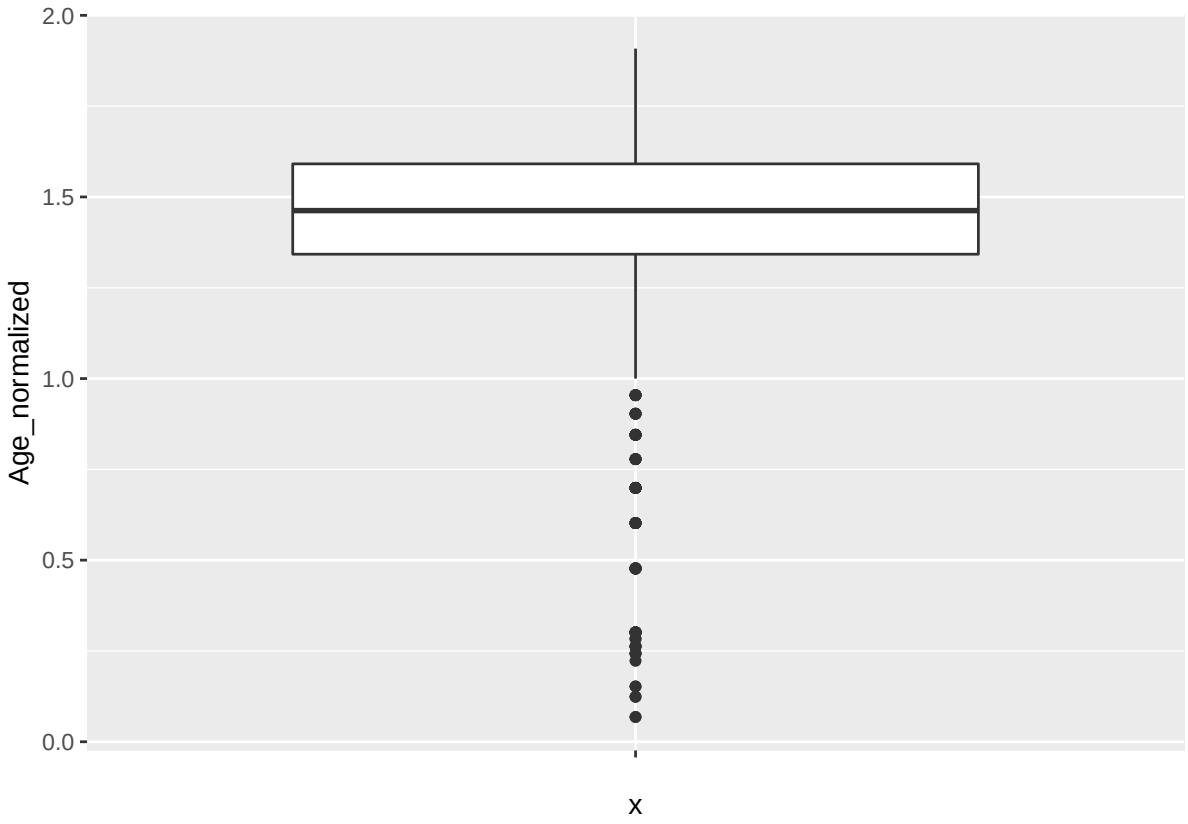

```
Age_boxplot
```



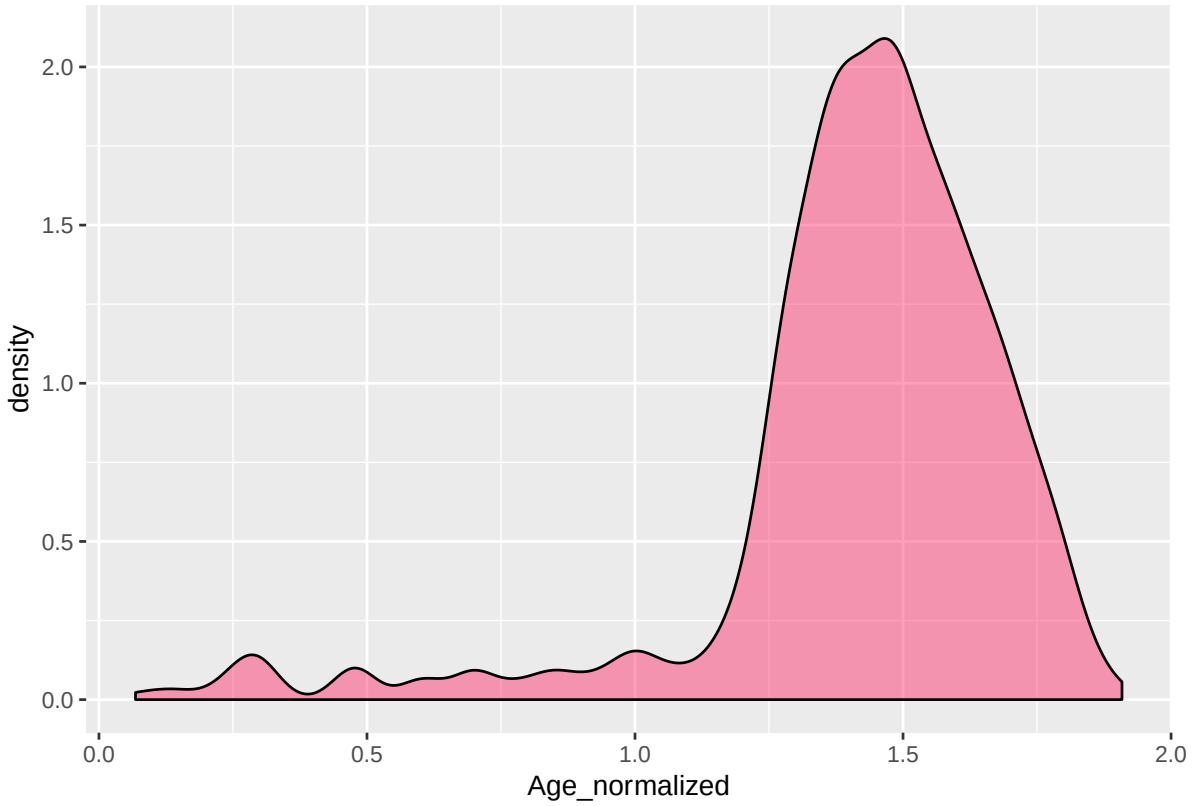
De la gráfica, observamo que la mayoría de la población se encuentra entre 0 y 60 años aproximadamente. Pero hay pasajeros que se encuentra entre los 60 y los 80 años. Por lo que no parece que hay errores tipográficos.

Pero ese grupo de valores extremos entre 60 y 80, pueden

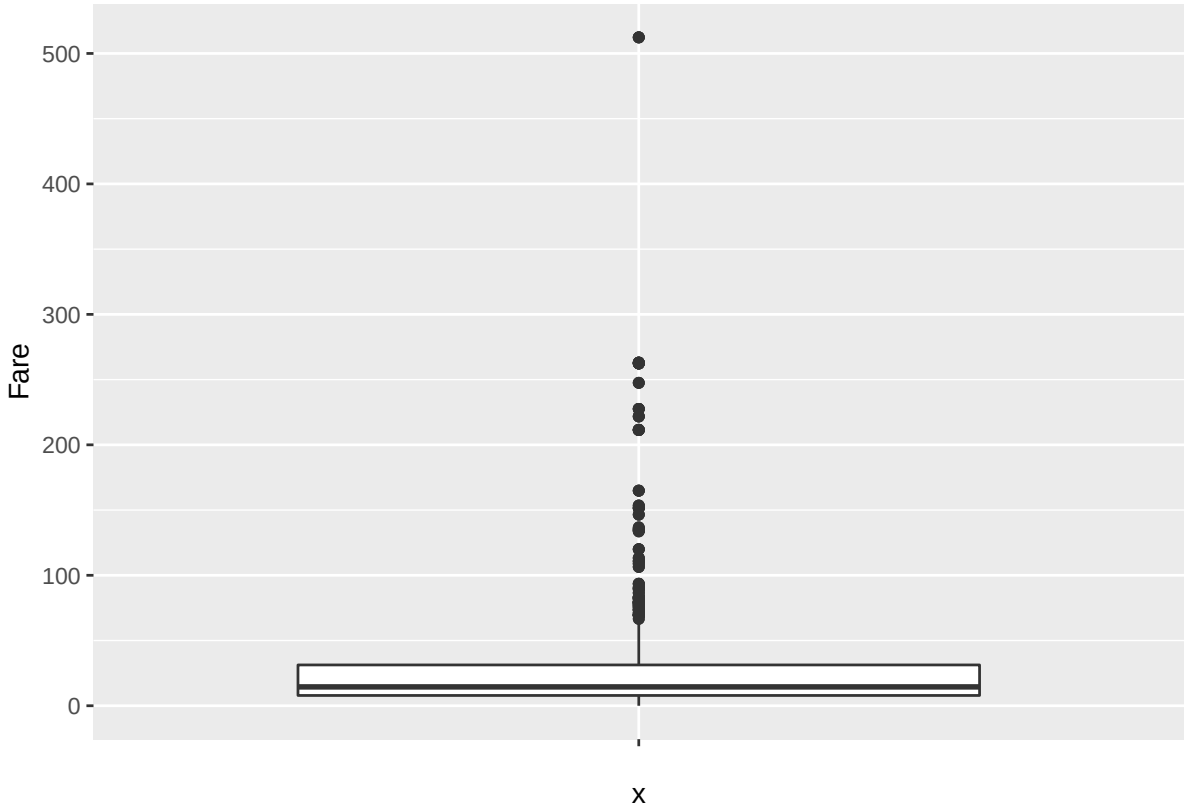
```
Age_normalized<- log10(titanic$Age +1 )
Age_normalized_boxplot <- ggplot(titanic, aes(x="", y=Age_normalized) ) +
  geom_boxplot()
Age_normalized_boxplot
```




```
Age_normalized_density <- ggplot(titanic,
  aes(x = Age_normalized)) +
  # Función de densidad de los valores de Age filtrados
  geom_density(fill = '#ff0f55', alpha=0.4, na.rm=T)
Age_normalized_density
```




```
Fare_boxplot
```



De las observación de las gráficas, no podemos observar valores extremos que se puedan considerar erroneos

References