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

Autor: Giovanny Caluña y Ivan Ovalle

Mayo 2021

Contents

Detalles de la actividad

Descripción

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

Objetivos

Los objetivos que se persiguen mediante el desarrollo de esta actividad práctica son los siguientes: - Aprender a aplicar los conocimientos adquiridos y su capacidad de resolución de problemas en entornos nuevos o poco conocidos dentro de contextos más amplios o multidisciplinares. - Saber identificar los datos relevantes y los tratamientos necesarios (integración, limpieza y validación) para llevar a cabo un proyecto analítico. Aprender a analizar los datos adecuadamente para abordar la información contenida en los datos. - Identificar

la mejor representación de los resultados para aportar conclusiones sobre el problema planteado en el proceso analítico. - Actuar con los principios éticos y legales relacionados con la manipulación de datos en función del ámbito de aplicación. - Desarrollar las habilidades de aprendizaje que permita continuar estudiando de un modo que tendrá que ser en gran medida autodirigido o autónomo. - Desarrollar la capacidad de búsqueda, gestión y uso de información y recursos en el ámbito de la ciencia de datos.

Competencias

Las competencias que se desarrollan son:

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

Desarrollo

Descripción del dataset

Para el desarrollo de este proyecto se seleccionó el data set llamado: Titanic: Machine Learning. Obtenido de: (https://www.kaggle.com/c/titanic). El data set cuenta con la informacion de diferentes tripulantes que estuvieron en el Titanic el dia de su hundimineto. El data set cuenta con 12 atributos que ayudan a describir a cada pasajero: - PassengerID: Id del pasajero registrado. - Survived: Si el pasajero sobrevivió o no al incidente(0 = No, 1 = Si). - Pclass: Clase del ticket abordo del Titanic. Ej: 1,2 o 3. - Name: Nombre del pasajero. - Sex: Genero del pasajero. - Age: Edad del pasajero. - SibSp: Número de hermanos o conyuges a bordo del Titanic. - Parch: Número de padres o niños a bordo del Titanic. - Ticket: Número del ticket. - Fare: Tarifa del pasajero (costo del ticket). - Cabin: Número de cabina abordo del titanic. - Embarked: Puerto de embarcación (C = Cherbourg, Q = Queenstown, S = Southampton)

Importancia y objetivos de los análisis

Tomando en cuenta la informacion que nos provee el data set, el siguiente trabajo tratará de desarrollar un modelo de clasificacion supervisado XXXXXXXXX, que nos ayudará a predecir si un pasajero sobrevivió o no al hundimiento del Titanic, basado en sus atributos. Este modelo nos ayudara a ratificar o contrastar las diferentes hipotesis que se deprenden en el analisis visual de los datos.

Limpieza de los datos

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

Eliminación de atributos

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

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

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

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

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

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

```
summary(passengers)
```

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

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

Para la eliminacion de atributos innecesarios utilizaremos la funcion *str* de R que nos dara diferente ejemplos de cada atributo asi como su formato.

```
str(passengers)
```

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

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

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

Normalizacion de datos

Como se pudo observar en el apartado anterior, solo contamos con un atributo (fare) de tipo numeric, que se puede normalizar para eliminar o disminuir grandes cambios debido a la diferencia entre valores del mismo atributo.

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

Valores nulos

Ahora procederemos a tratar los valores nulos. Primero aplicamos el siguiente comando que nos ayudara a contabilizar los valores nulos de cada atributo:

```
colSums(is.na(passengers))
```

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

Con la funcion is.na podmeos observar que el data set cuenta con varios datos vacios o nulos en el atributo Age (edad). Debido al alto numero de valores nulos y a su alta relevancia para el analisis, vamos a analizar de manera detenida el atributo. Con el siguiente comando, vamos a ordenar los valores que toma el atributo basado en el numero de apariciones en el data set.

Como se puede observar, en la primera fila (los valores que mas se repiten), se encuentran valores en el rango de 20 a 30 años aproximadamente. Para poder visualizar la infomracion de manera grafica, cargamos las librearias necesarias y procedemos a plotear la informacion con el siguiente comando:

```
if(!require(ggplot2)){
    install.packages('ggplot2', repos='http://cran.us.r-project.org')
    library(ggplot2)
}
```

```
## Loading required package: ggplot2
```

```
if(!require(ggpubr)){
    install.packages('ggpubr', repos='http://cran.us.r-project.org')
    library(ggpubr)
}
```

```
## Loading required package: ggpubr
```

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

```
if(!require(grid)){
    install.packages('grid', repos='http://cran.us.r-project.org')
    library(grid)
}
```

```
## Loading required package: grid
```

```
if(!require(gridExtra)){
    install.packages('gridExtra', repos='http://cran.us.r-project.org')
    library(gridExtra)
}
```

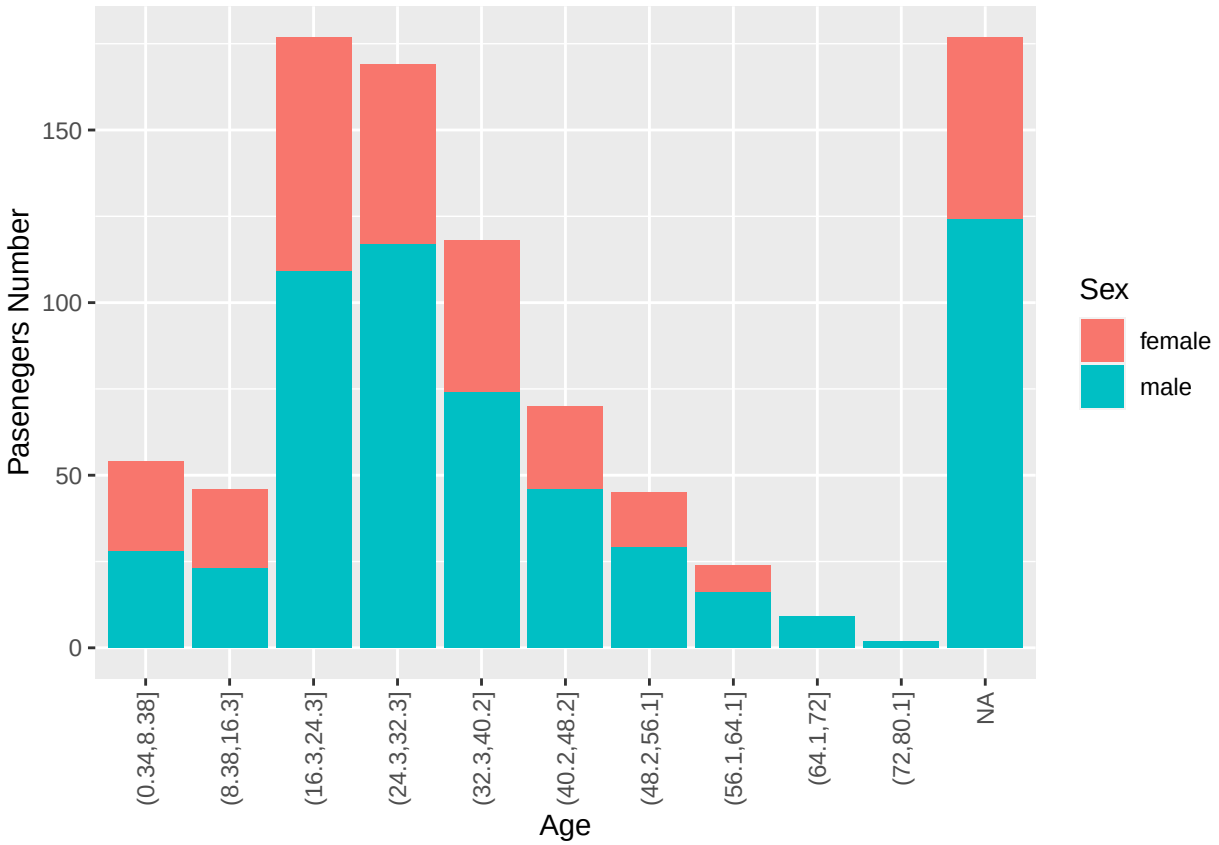
```
## Loading required package: gridExtra
```

```
if(!require(C50)){
    install.packages('C50', repos='http://cran.us.r-project.org')
    library(C50)
}
```

```
## Loading required package: C50
```

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

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

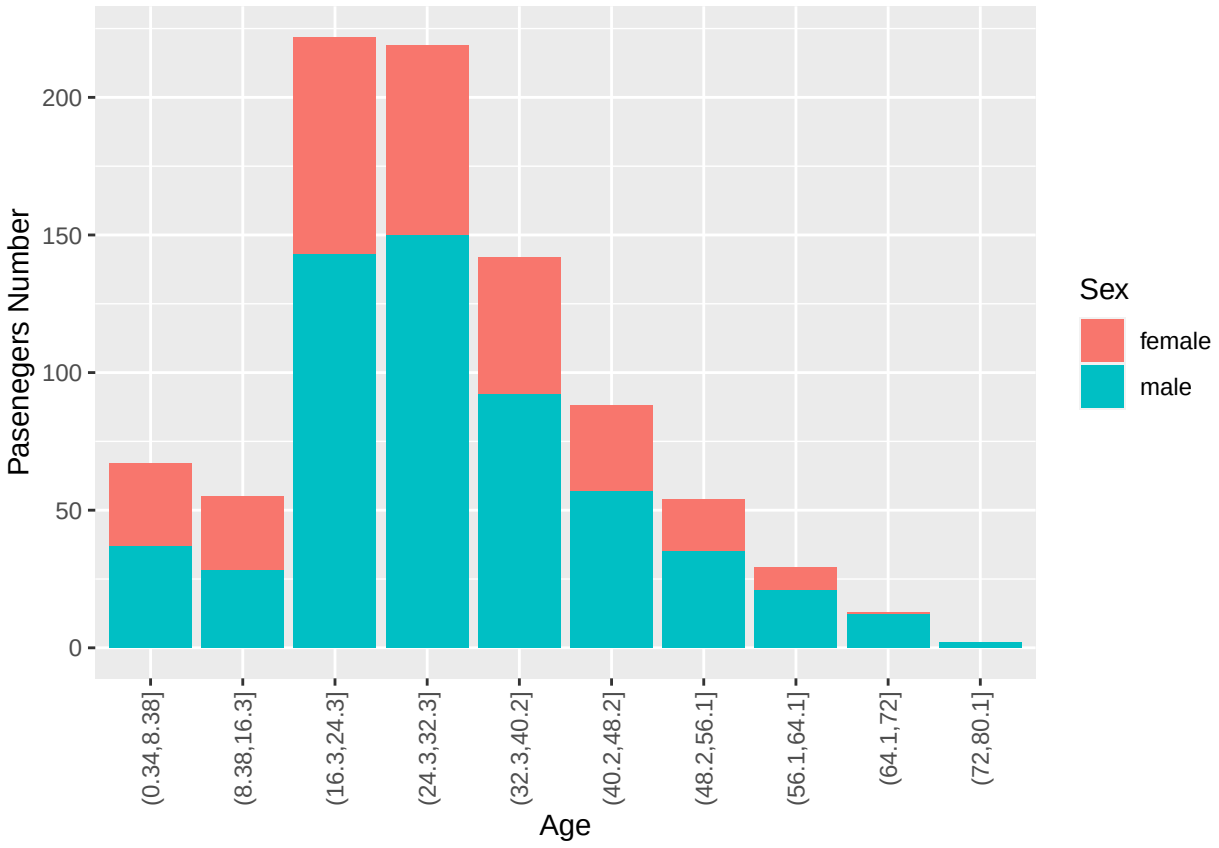


Como se puede observar en el gráfico, la mayor parte de pasajeros se encuentra distribuido en edades que van desde los 16.3 años hasta los 32.3. Ademas, podemos apreciar, el numero de hombres y mujeres mantiene la misma proporcion en los todos intervalos. Por esta razon, se procede a llenar los datos faltantes del atibuto edad con valores random en el intervalo con mayor numero de pasajeros, sin distincion si es hombre o mujer.

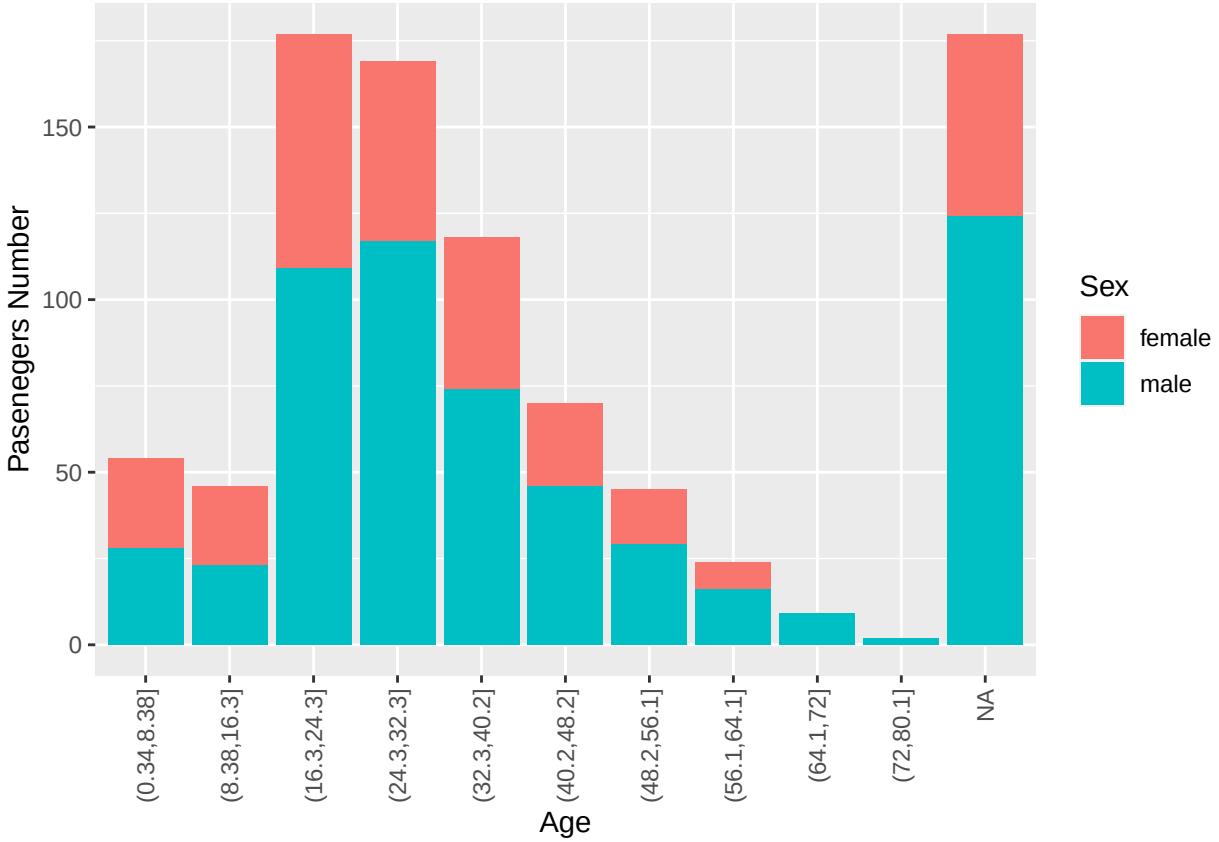
Para reemplazar los valores nullos, primero creamos una columna con todos los valores

```
set.seed(873465)
tmpAge <- passengers$Age
tmp <- passengers$Age[!is.na(passengers$Age)]


passengers$Age[is.na(passengers$Age)]  <- sample(tmp, size = 177, replace = FALSE)
```

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




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



Ahora procedemos a normalizar el atributo que contiene la tarifa del pasajero (Passenger Fare)

Transformacion de Box Cox “

```
library(missForest)
```

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

## Loading required package: randomForest

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

## randomForest 4.6-14

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

##
## Attaching package: 'randomForest'

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

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

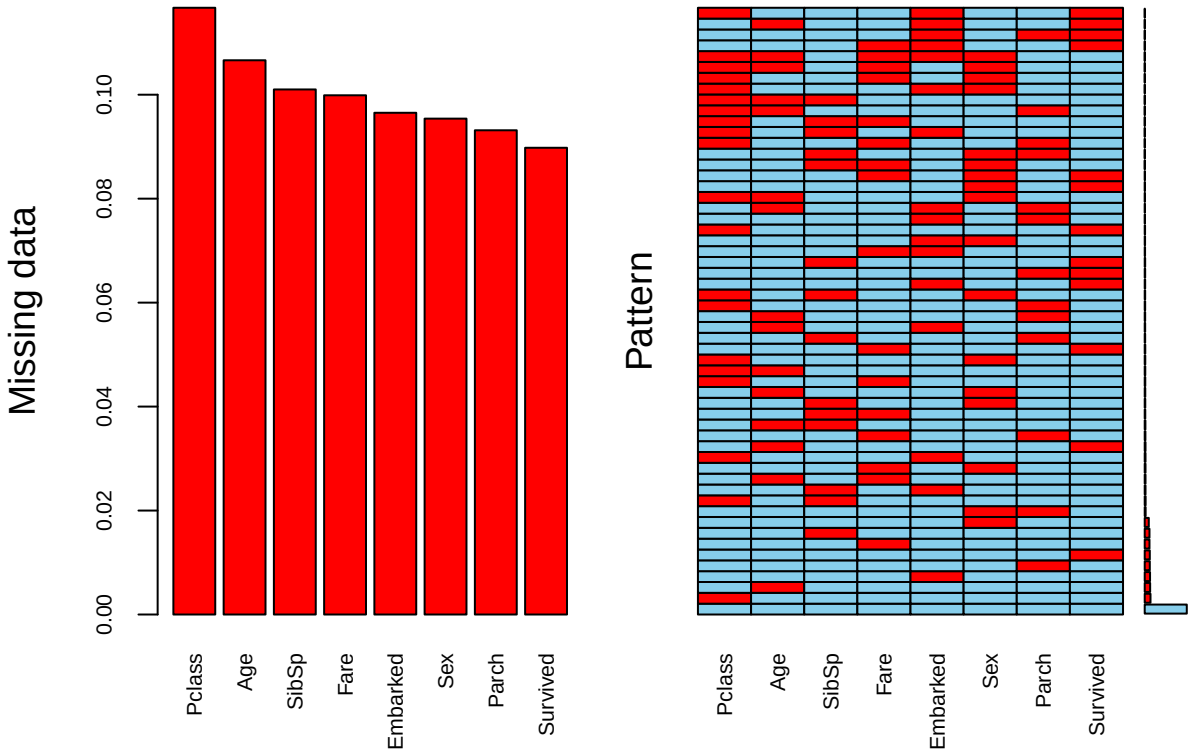
```
## Loading required package: foreach

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

## Loading required package: itertools

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

## Loading required package: iterators

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

```
library(VIM)
```

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

## Loading required package: colorspace

## VIM is ready to use.

## Suggestions and bug-reports can be submitted at: https://github.com/statistikat/VIM/issues

##
## Attaching package: 'VIM'

## The following object is masked from 'package:missForest':
##
##     nrmse

## The following object is masked from 'package:datasets':
##
##     sleep
```

```
passengers.mis <- prodNA(passengers, noNA = 0.1)
aggr(passengers.mis, numbers=TRUE, sortVars=TRUE, labels=names(passengers.mis),
cex.axis=.7, gap=3, ylab=c("Missing data","Pattern"))
```

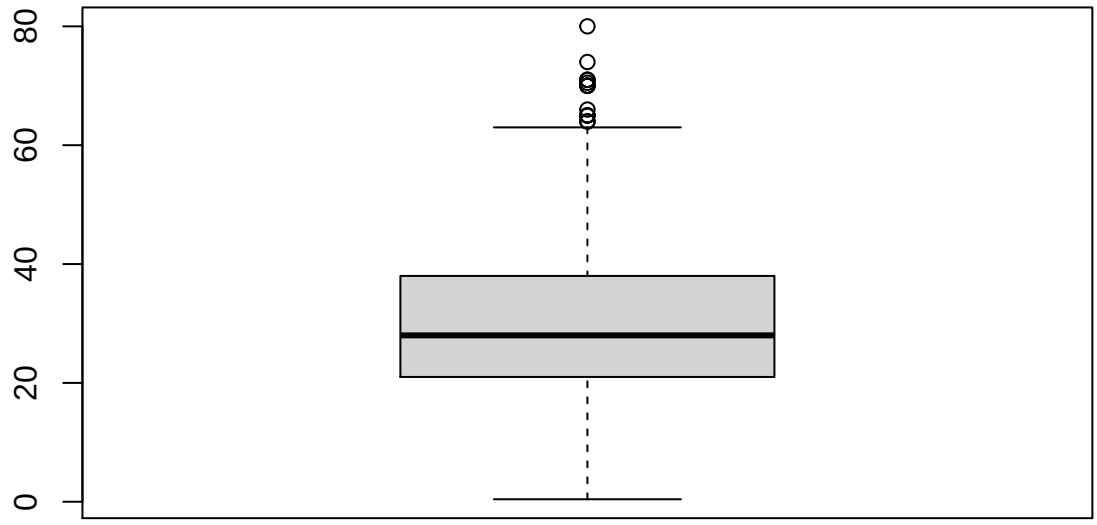
```
## Warning in plot.aggr(res, ...): not enough vertical space to display frequencies
## (too many combinations)
```



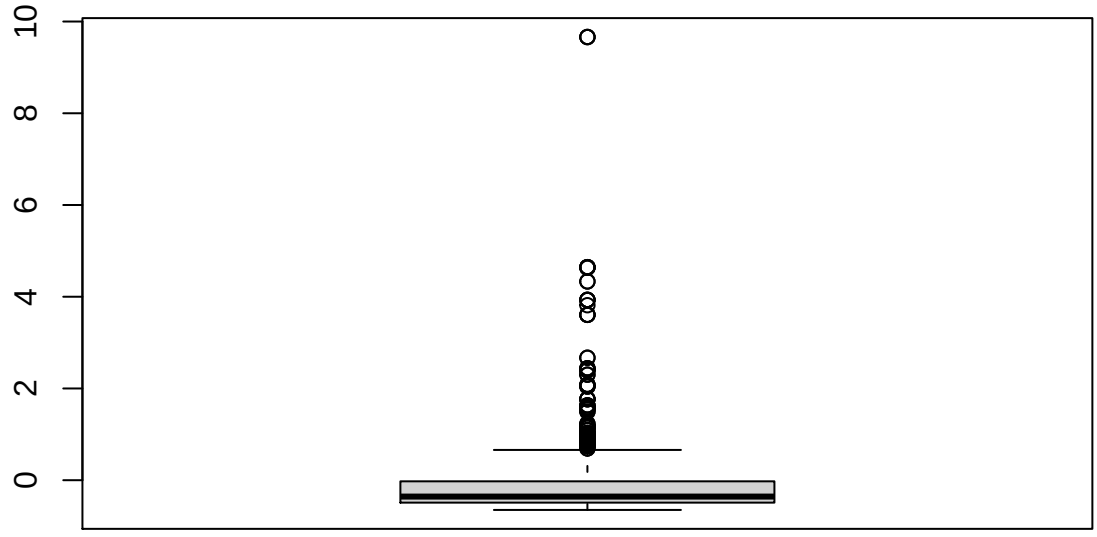

```
##
##  Variables sorted by number of missings:
##  Variable      Count
##    Pclass 0.11672278
##       Age 0.10662177
##     SibSp 0.10101010
##      Fare 0.09988777
##  Embarked 0.09652076
##       Sex 0.09539843
##     Parch 0.09315376
##  Survived 0.08978676
```

oOUtliers

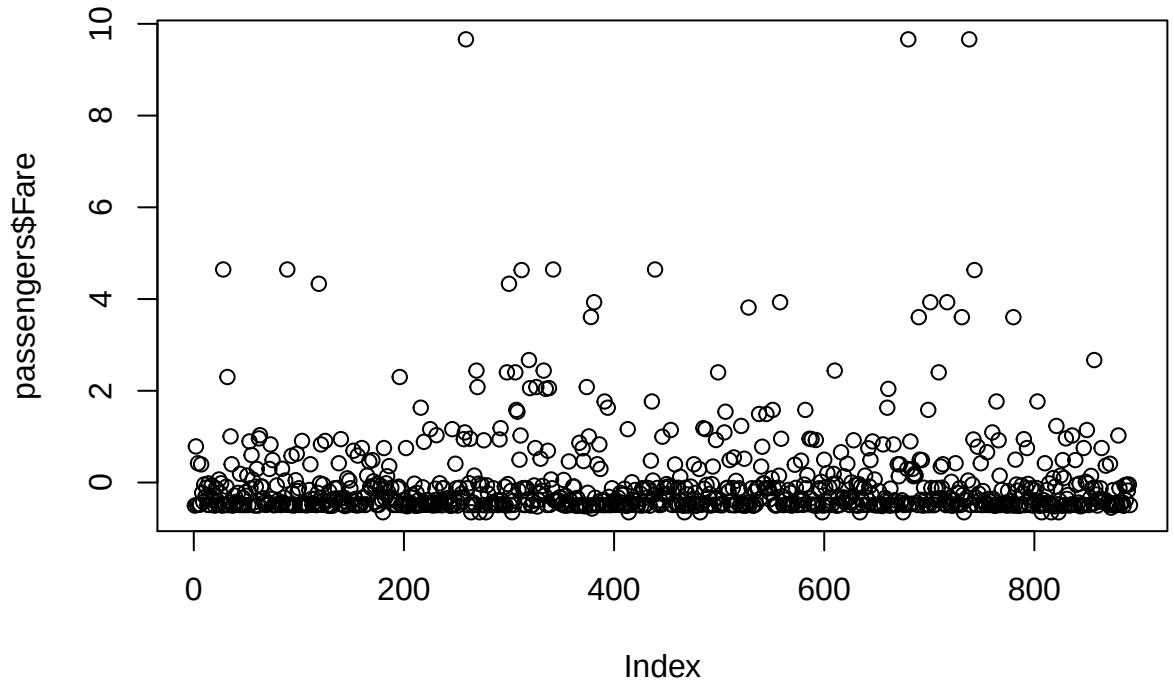
```
AgeOutliers<-boxplot(passengers$Age,main="Sepal Width")
```




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




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




```
#Calcular outliers mas de 2 dvs

sd(passengers$Fare)
```

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

```
passengers.FareOutlier <- abs(passengers$Fare) > 3

#plot(passengers$Fare, pch=passengers.FareOutlier)
```

```
#passengers.pca <- prcomp(passengers[,c(1:3,5:8)],na.rn=TRUE, center = TRUE, scale = TRUE)
#summary(passengers.pca)
```